A NEURAL NETWORK FACE DETECTOR DESIGN USING BIT-WIDTH REDUCED FPU IN FPGA

A Thesis Submitted to the College of

Graduate Studies and Research

In Partial Fulfillment of the Requirements

For the Degree of Master of Science

In the Department of Electrical and Computer Engineering

University of Saskatchewan

Saskatoon

By

Yongsoon Lee

PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Electrical and Computer Engineering,

57 Campus Drive,

University of Saskatchewan,

Saskatoon, Saskatchewan,

Canada S7N 5A9

ABSTRACT

This thesis implemented a field programmable gate array (FPGA)-based face detector using a neural network (NN), as well as a bit-width reduced floating-point unit (FPU).

An NN was used to easily separate face data and non-face data in the face detector. The NN performs time consuming repetitive calculation. This time consuming problem was solved by a Field Programmable Gate Array (FPGA) device and a bit-width reduced FPU in this thesis. A floating-point bit-width reduction provided a significant saving of hardware resources, such as area and power.

The analytical error model, using the maximum relative representation error (MRRE) and the average relative representation error (ARRE), was developed to obtain the maximum and average output errors for the bit-width reduced FPUs. After the development of the analytical error model, the bit-width reduced FPUs and an NN were designed using MATLAB and VHDL. Finally, the analytical (MATLAB) results, along with the experimental (VHDL) results, were compared. The analytical results and the experimental results showed conformity of shape.

It was also found that while maintaining 94.1% detection accuracy, a reduction in bit-width from 32 bits to 16 bits reduced the size of memory and arithmetic units by 50%, and the total power consumption by 14.7%.

ACKNOWLEDGEMENTS

I thank my advisor, professor Seok-Bum Ko. He showed deep patience and endurance on my research, allowed me to study through many possible experiments so that I could make this achievement.

I would like to thank professor Hsiang-Yung Daniel Teng and Anh Van Dinh at the University of Saskatchewan for teaching me about embedded systems. I wish to thank thesis committee members, Professors, Ronald J. Bolton, Carl McCrosky, and Derek Eager who monitored my work and provided valuable comments.

I cannot forget the precious help from other professors and staffs including the writing and math help centers at the University of Saskatchewan. Their efforts helped my research to be conducted much more smoothly.

I thank my co-workers: Alex, Dongdong, Huy, Rouf, Son, and Song at the VLSI Labs and Jaydrath at Bio-medical engineering. They inspired me through discussion; we discussed not only research, but also life and personal concerns.

I also thank all of my friends: Shirley, Jerry, Serra, Tim, Ken and Ann, and my church. Their warm concern helped me to feel comfortable and to settle down in Canada.

I am grateful to my parents, brother, sister, and nephews for their endless love and sacrifice.

Finally, I would like to give my special thanks to my wife, Hyeyoung. I could not have done anything without her devoted help.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ARRE	Average Relative Representation Error
EER	Equal Error Rate
FAR	False Acceptance Rate
FFNN	Feed Forward Neural Network
FPGA	Field Programmable Gate Array
FPU	Floating-Point Unit
FRIDGE	Fixed-point Programming Design Environment
FRR	False Rejection Rate
FXU	Fixed-Point Arithmetic Unit
HDL	Hardware Description Language
ISE	Integrated Software Environment
LUT	Look-Up-Table
MAC	Multiplication and Accumulation
MLBP	Multi-Layered Back Propagation
MLP	Multi-Layer Perceptron
MRRE	Maximum Relative Representation Error
NN	Neural Network
XST	XILINX Synthesis Technology

CHAPTER 1 INTRODUCTION

A recognition system is used in order to automatically recognize the objects. This automatic object recognition is mainly used for automation and security systems. Representative examples are automation machinery in factories, robots, and guided missiles.

A recognition-based security system decides whether an object passes or not. Security systems based on recognition of our physical characteristics allow operations that are independent of security cards and passwords. Security recognition systems that will utilize physical characteristics such as a person's face, fingerprints, and iris' have been mainly considered for the security system called "Biometric systems".

Iris and fingerprint recognition systems have been used for high security systems due to their high recognition rates. The disadvantage of both recognition systems is that the sensor needs to touch the object directly or to scan a person's eye by a laser beam which makes people fearful. Face recognition systems are convenient due to their non-invasive methods. Another advantage of face recognition is the object's possible unawareness of the process. Therefore, the system can be used to search objects in wide areas and to stealthily detect suspicious people.

Due to the low recognition rates of face recognition algorithms, face detection is more frequently used in the commercial market rather than in security recognition systems.

Among the face detector algorithms, an NN was used in this thesis because the NN is useful to classify the complex data like face data. Another advantage of the NN is that it is suitable to design in hardware based embedded systems due to its simple and repetitive calculations. A neural network (NN) detector can be used in various applications such as an automatic number plate recognition, postal code recognition, and face recognition systems. Because of the repetitive calculation, the NN has some problems in high speed detection. A dedicated embedded system using an FPGA and a bit-width reduced FPU solves the operating speed problem of the NN.

A floating-point unit (FPU) was developed due to its advantage and wide range of scales. The primary disadvantage of FPUs is their complexity. This problem can be addressed by using a bit-width reduced FPU. In order to decide the number of bits of an arithmetic unit, we need to know how many bits are required by the NN-based face recognition problem. To decide the required number of bits in FPU, an error model and design environment for an NN and a bit-width reduced FPU was developed. The error model was helpful to estimate the algorithm performance like detection rate error, and the hardware specification of area and speed. The design environment co-working of MATLAB and VHDL was developed to design and verify the VHDL program.

1-1 Motivations

1-1-1 Face Detection and Recognition System

The face recognition system is more convenient than other recognition methods because it is non-invasive and at times, the targeted object may be unaware of the process However, in terms of recognition performance, face recognition systems have a much lower recognition rate, between approximately 60 ~ 80% [1].

Face detection systems are more practical than total recognition systems [2, 3]. For example, the face detector system can be helpful in detecting suspicious faces, and manual check in an airport or big stadium where there are many people and it is very difficult to manually check. Therefore, face detection was considered and designed in this thesis.

1-1-2 Hardware based Fast Detector

The disadvantage of a software based detector is its slow processing time. Fast detectors can be implemented by a hardware-based system. The hardware based detectors have two advantages: fast speed and reliability in terms of scheduling, as compared to a general computer system.

Image processing is a time consuming process because of having to handle a big data size. A general computer camera is supported by 800 x 600 size images, and some image processing, like data format conversion (for example, from RGB color to YCbCr color), normalization, histogram equalization, filtering and other image enhancement process, are normally required for any image pre-processing [4]. (Note that recent computer

monitors use a larger size data to increase the image size and monitor size. The main limitation of the monitor and the PC camera size is not sensor size, but the time consuming problems caused by interface or image processing.)

The Following describes how many time units are required for normal image processing:

For mono color, low pass filtering (LPF) by mask, even assuming fast algorithm like 3 x 3 masking, takes 4,320,000 calculation times are required for one process.

(image size: 800 x 600) x 1(mono color) x 9(filter mask) =4,320,000 times.

Assuming the clock is 132MHz, 32.8ms is required for one process. 132MHz clock frequency is recently used for embedded graphic processor [5].

$$4{,}320{,}000 \times 7.6\text{ns}(132\text{MHz}) \approx 32.8\text{ms}$$

If 5 processes are applied to image processing, assuming each processing has the same masking time, 164ms is required.

$$5 \text{ process} = 32.8\text{ms} \times 5 = 164\text{ms}$$

It means about 6 frames can be treated per second.

Therefore, it is obvious that high speeds allow for more image enhancement processing.

Even commercial software based face detector programs have speed problems, taking 1 second to detect 1 frame. There are two main reasons for the speed problems of software based detector: interface and scheduling of software.

First of all, one of time consuming sources of a software based system is the interface. The interface between PC and sensor (camera) is a time consuming process due to arbitration and transfer time.

To reduce the transfer time, in other words to raise the frame rate, two methods are used for a general computer: reduction of image data size and JPEG compression. Both of the methods are data loss compression methods.

Figure 1-1 shows the maximum image size and frame rate for a normal PC camera. To maintain the 30 frame, 352 by 288 are the maximum image size. Therefore, to maintain 800 by 600, frame rates will be reduced to 15 frames per second or less.

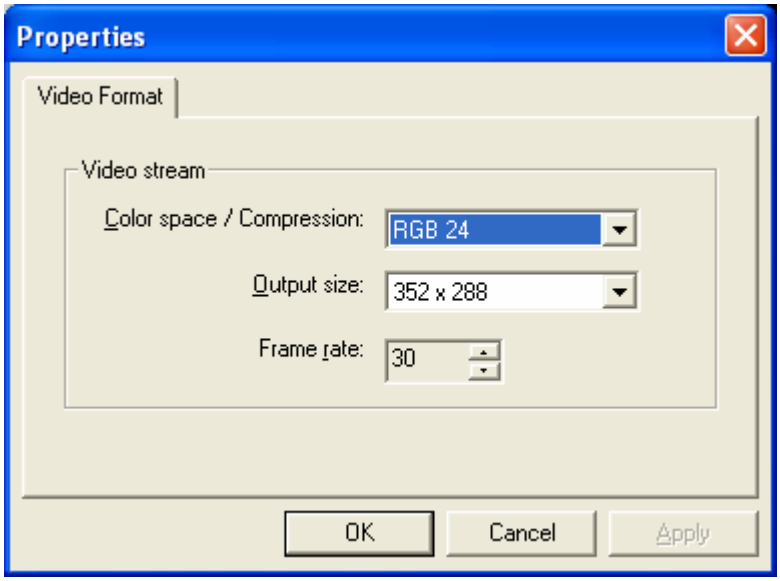
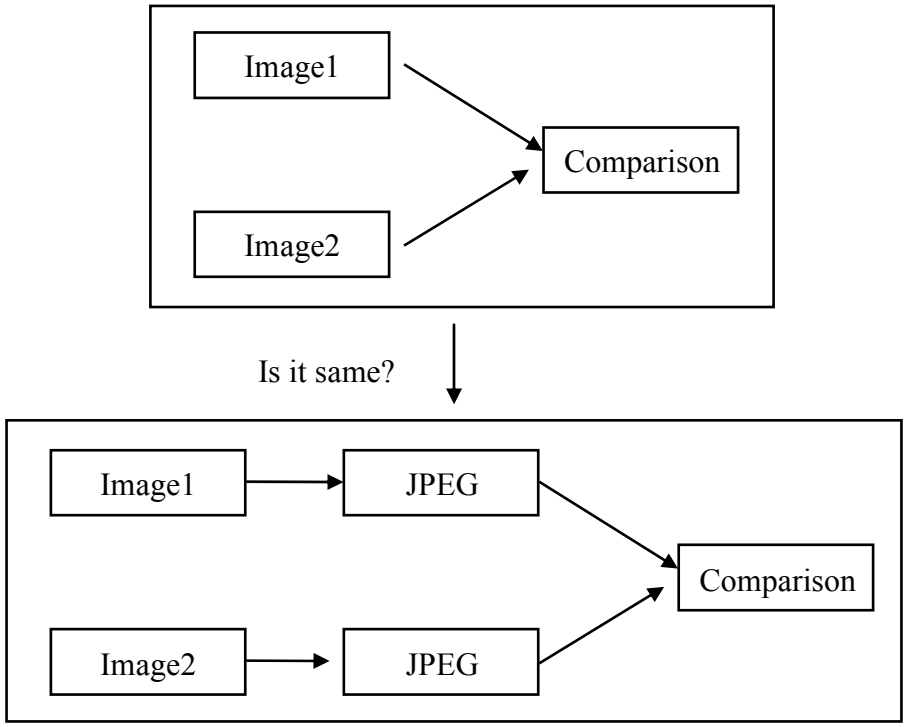


Figure 1-1 Max image and frame size of normal PC camera

Figure 1-2 shows how the JPEG loss compression method affects recognition performance. Correlation was used to compare similarity of two images. JPEG compression reduces the correlation value from -0.0795 to -0.0817 due to a data loss of compression.

Two loss compression methods reduce the recognition performance. Therefore, a higher speed operation allows more data so that it improves the recognition performance.



(a) Experiment diagram

(b) Correlation result of BMP and JPEG image

Figure 1-2 Correlation difference affected by JPEG loss compression

Another time consuming source of a software based system is Windows OS scheduling. The scheduling also affects the systems reliability because of the difficulty in estimating the processing time. When software based systems delay the image processing

time, software based detector can miss the chance to detect an object when it is moving fast. This is a serious problem for security.

Standard environments of a general computer, like interfacing with HDD and memory, also increase the detection time.

Therefore, H/W (detection) + S/W (recognition) can improve the performance like processing time and reliability in image processing. This separated system can save the S/W effort and burden.

Figure 1-3 shows an example of the type of separated system by detection and recognition functions [6]. The recognition part can be accessed through different devices such as servers, PCs, mainframes, etc.

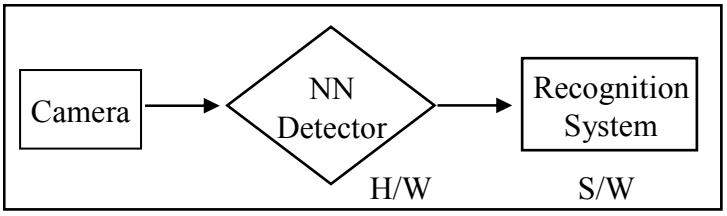


Figure 1-3 Example of face detection system

A computer will display the images we need to watch. We do not have to watch every spot at every second, like Figure 1-4. Only detected objects from the camera will be sent to the main server. Note that the camera has a hardware based detector. This saves management costs and reduces the burden on the server or main computer. Furthermore, faster detection will capture more frames and frames of higher quality.

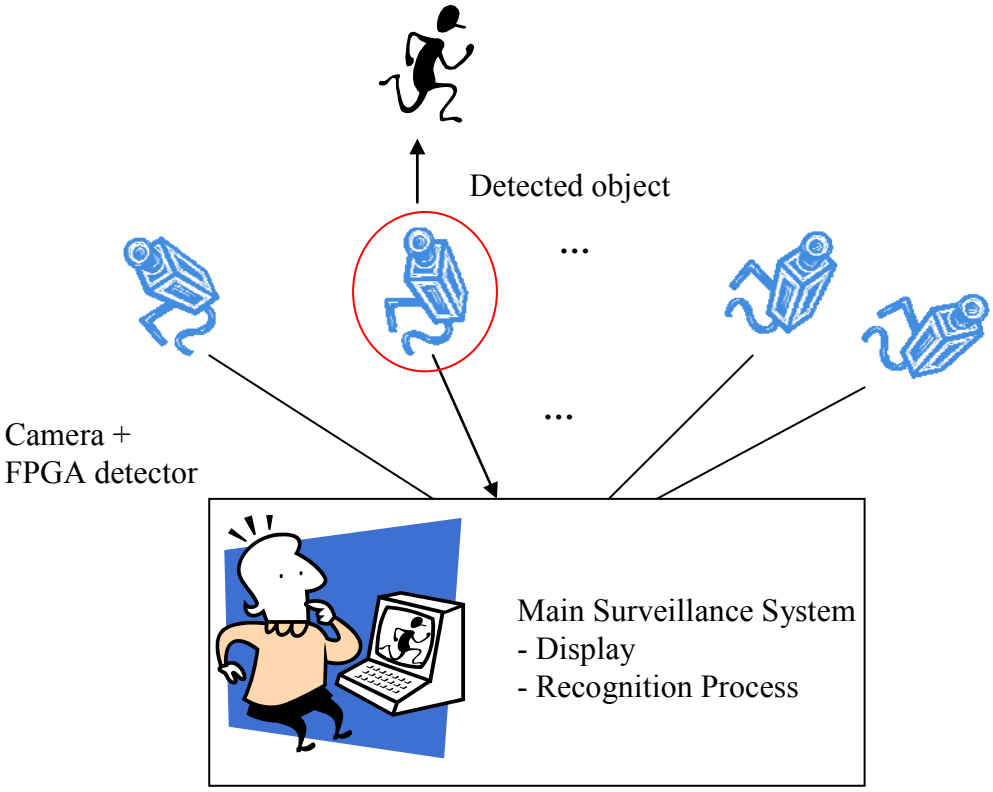


Figure 1-4 Example of building surveillance system

Hardware based detectors save the image processing time and contributes to the reliability of the system by reducing extra work, like interfacing and scheduling of S/W.

Therefore, the hardware based system is attractive for image processing systems which require huge data processing and high resolution. The main reason why a hardware based graphic card is required for image processing is because it improves image processing time and image resolution. Chapter 5 compares the processing time of software and hardware based FPGA detector.

1-1-3 Face Detector based on a Neural Network

The NN is useful in classifying complex data, like face data, and suitable to design in the hardware based embedded system with simple and repetitive calculation.

The third advantage of an NN detector is strong for the change of environment. When the environment of the NN detector is changed, the performance can be maintained through the changes of weights. For example, if the NN detector was fitted to run under a bright environment, the performance can be decreased under a dark environment. In this case, weights data can be renewed after learning the procedure with the new database. The weights data, in other words, coefficients of the NN, are obtained through the learning procedure of the NN.

Therefore, the same detectors can be used in various environments and conditions. This flexibility of the NN is an attractive feature for a designer.

A neural network (NN) detector can also be used not only for face detectors, but also for recognition system or various applications such as car inspections and postal code extractors.

1-1-4 Neural Network Face Detector based on FPGA

Even though the NN is simpler than other algorithms, the complexity of the NN through repetitive calculation is still the major issue to be solved.

Windows and computer based detectors take 1 second to detect a 1 frame image according to experiments with the commercial product. Scheduling of a Windows operating system (OS), communication time from the device using USB or LAN, and hardware communication time such as loading time from the file, HDD or Memory, causes the system to work slowly.

Hardware based, dedicated embedded system and bit-width reduced FPU can solve this speed problem.

Among some devices for dedicated hardware based embedded system such as a digital signal processor (DSP), a microprocessor like an advanced RISC machine (ARM) and a field programmable gate array (FPGA). An FPGA was considered as a device for the NN design due to its high speed by both pipelining and flexible circuits. A DSP was developed for digital signal processing and a microprocessor is used for a small-size embedded system, which is not required of high speed performance.

An FPGA has no limit for pipelining, while DSP has support for 4 pipelines which can calculate 4 calculations at the same time. Moreover, using an FPGA allows the designer or researcher to design a flexible circuit, like a bit-width reduced FPU.

Therefore, the detector based on an FPGA provides flexibility by allowing the system to work faster. Moreover, an FPGA has many I/Os (Inputs and Outputs) which allow for a sensor system to have many sensors. For example, XC3S4000 has 250 I/Os and DSP has 32 I/Os (TI TMS320C6701 floating-point DSP).

Moreover, some FPGA's include a microprocessor and a DSP module inside so that an FPGAs will be the most popular device [7].

1-1-5 Bit-Width Reduced FPU for Embedded System

In order to design an embedded system for specific applications, we need to decide the proper arithmetic unit for calculation. Among representative arithmetic units, a fixed-point unit (FXU) and a floating-point unit (FPU), an FXU is simpler than an FPU and this makes a system simple and work fast. However, the disadvantage of an FXU is that it has a smaller range than an FPU so a designer needs to consider the exact number range of a digital system to avoid overflow or underflow error.

The advantage of an FPU is that it has a wide range compared to an FXU. Therefore, a digital system can be easily designed with an FPU with less concern about range than an FXU. Data range can be easily changed without changes of the number system.

The feature of wide range is also effective for NN systems because the NN requires the wide range when the learning weight is calculated or changed [8, 9].

Another advantage is ease of use. A digital system can share the number system with other digital systems. For example, there is a standard for the FPU, and most computers use an FPU, so that a program in a computer can be applied for a digital system having an FPU without changing the number system. These advantages are the main reason why we use the FPU on our personal computer.

These two reasons, wide range and ease of use, are the main reasons why general computer use the FPU.

Floating-point hardware offers a wide dynamic range and high computation precision, but it occupies a large size of the chip area and consumes more energy. Therefore, its usage is very limited. Many embedded microprocessors do not even include a floating-point unit due to their unacceptable hardware costs.

A bit-width reduced FPU solves this complexity problem. A floating-point bit-width reduction can provide a significant saving of hardware resources such as area and power.

It is useful to understand the loss in accuracy and the reduction in costs as the number of bits in an implementation of floating-point representation is reduced.

Incremented reductions in the number of bits used can produce useful cost reductions, while still meeting any given accuracy requirement.

Flexible bit-width reduced FPU IP is useful not only for study, but also for the industry because there is no free commercial Intellectual Property (IP). Moreover, there is no simulation environment model based on bit-width reduced FPU to help with verification before the chip is designed.

There are several tools for the FXU number system simulation, such as FRIDGE (Fixed-point Programming Design Environment), MATLAB program language and a FPU to FXU converter program to simulate the performance of finite precision FXU [10, 11, 12]. However, not many programs are available for a reduced precision FPU.

FPGA vendors provide some FPU IP through third party companies shown in Table 1-1. Most companies provide FPU multiplication, addition, and some operations following the standard which has 32 bits for single precision, 64bits for double precision and an extended mode. There is no free commercial bit-adjustable FPU addition and multiplication IP.

Table 1-1 FPU IPs of XILINX and ALTERA [13, 14]

	3rd party	free	Operation	bits	Stages	Language
XILINX	Qinetiq	-	×, /,+,-,root ,compare	32,64 (extended: 43,79) Scalable (12~64bits)	3,4,5	VHDL
ALTERA	Amphion	Multiplication (32,64,43 bits)	+,×,/	32,64, (extended :43)	5,6 (for ×)	VHDL

1-1-6 Finite Precision Error Analysis

Having an adjustable bit in circuits is an easy way to optimize the circuit reducing extra hardware cost and for increasing the hardware performances like area and speed. To determine the required number of bits in the bit-width reduced FPU, the error caused by a reduced precision needs to be analyzed. Developed error analysis including FPU accuracy indices, MRRE, and circuit errors are helpful to estimate the upper bound error of applications.

Reduced precision error analysis for NN implementations was introduced in [15]. A formula that estimates the standard deviation of the output differences of fixed-point and floating-point networks was developed in [16]. Previous error analyses are useful to estimate possible errors. However, it is necessary to know the maximum and average possible errors caused by a reduced precision FPU for practical implementation.

Therefore, in this thesis, the error model was developed using the maximum relative representation error (MRRE) and average relative representation error (ARRE) which are representative indices to examine the FPU accuracy. One more error term caused by the arithmetic unit is the rounding error of arithmetic calculation which was added in result in contrast to an input quantization error [17]. After the error model for the reduced precision FPU was developed, the bit-width reduced FPUs and the NN were designed using MATLAB and VHDL. Finally we compared the analytical (MATLAB) results with the experimental (VHDL) results.

The design environment co-working of MATLAB and VHDL was developed and it was also very helpful to design and verify the VHDL program. Reliability is the

important factor for arithmetic. A test with all possible numbers using the manual is very difficult; therefore, the automatic test program environment provides reliability.

The MATALB program is a popular and familiar program tool for an engineer and it was used to program the NN face detector, to provide test-bench for verifying VHDL program and to analyze the output error caused by bit-width reduced FPU and the NN circuit.

1-2 Contributions

1-2-1 Fast Neural Network Detector based on an FPGA device

Hardware based dedicated embedded detectors using an FPGA works faster than a software based system through the reduction of extra work, and it allows the user to process more frames by allowing more image processing algorithms, such as pre-processing and the enhancement algorithm.

The FPGA based detector can also be used as a stand alone type without a computer system. As a device, the FPGA was used due to its flexibility in circuit design, speed by pipeline, and its cheap cost.

1-2-2 Fast Neural Network Detector Using a Bit-Width Reduced FPU

A bit-width reduced FPU improves the hardware performance such as area, power, speed and chip cost. An adjustable bit FPU makes it easy to decide and design the dedicated embedded system.

1-2-3 Error Analysis and Design Environment

The numerical error model was developed to estimate the system specification through the estimation of the output error before we design the chip. MRRE and ARRE for bit-width reduced FPU were added in the error model to estimate the maximum output error and average output error. One more error term caused by the FPU circuit was added in the error model to be compensated for practical usage.

Moreover, the design environment co-working with MATLAB and VHDL make it easy to verify the circuit.

1-3 Outlines

This thesis is outlined as follows. In Chapter 2, the FPGA implementation of the NN face detector using the bit-width reduced FPUs is described. Chapter 3 explains how representation errors theoretically affect a detection rate in order to determine the required number of bits for the bit-width reduced FPUs. In Chapter 4, implementation methodology is described. In Chapter 5, the experimental results are presented, and then they are compared with the analytical results to verify if both results match closely. Chapter 6 draws conclusions.

CHAPTER 2 BACKGROUND

2-1 Face Recognition and Detection

People have the ability to recognize facial features and characteristics, and if a machine can do this work instead of people, it will save time and money.

Face recognition or detection is an identification or authentication process in various areas like security, retail, banking, industry and home. In order to let a machine automatically work by itself, the first job of the machine is to recognize the object.

Moreover, we are living in the ubiquitous era connected by wireless and complex network systems which require identification and authentication of an object far from the server. Therefore, authentication is very important for automatic and security systems.

The best advantage of face recognition compared with other biometrics, such as fingerprint and iris recognition system, is that it is easy to detect an object through a normal camera, which is a non-invasive process. Moreover, face recognition following people's recognition principle allows us to easily understand the recognition procedure and techniques. Another advantage of face recognition is that there are many database sources; for example, most identification cards include a picture of a person's face. The disadvantage of face recognition is that it has a low recognition rate and that it takes a long time to calculate the huge image data to increase the recognition rate.

The early technique of face recognition was started from the simple algorithm using basic facial features. Contemporaneously, many algorithms based on the statistics are

introduced. Examples of representative algorithms are principle component analysis (PCA), fisher linear discriminant (FLD), support vector machine (SVM), neural networks, and face recognition committee machine (FRCM) [2, 3, 18].

An NN has a good recognition rate, even though FLD, SVM, and FRCM are better [18]. Advantages of an NN are that it is easy to classify the complex data and to design hardware through simple processing elements, providing basic architecture of a multi-processor computer and neural computer systems. Moreover, the advancement of neuro-science is providing the opportunity to use biological neural networks to improve the performance to overcome the limitation of computational modeling based on statistics and mathematics.

Biometrics is the standard organization for the recognition system in the U.S [19]. NIST also invests and researches for the recognition system [20]. FERET database, supported by NIST, is widely used to test performance. Many institutions and universities like MIT, FERET, UMIST, University of Bern, Yale, AT&T (Olivettti), Harvard and Purdue opened face databases [3]. The database types are various in terms of the number of databases, the number of faces in a picture, color, angle of face, and pose. The proper database needs to be chosen for the applications. The Olivettti database was used in the thesis because of its advantages, like mono color, small size image, well organized data, and its free database were easy to use [3, 21].

2-2 Face Detection System based on a Neural Network

2-2-1 A Neural Network

A neural networks (NNs) system imitates the biological neural networks. The system makes it possible to classify data clusters using a learning procedure that utilizes examples, generalization, associative memory, and fault tolerance [22, 23].

The learning procedure is implemented through the mapping and grouping of a sufficiently large number of examples so that the system outputs a desired value. This process is composed of both supervised learning and unsupervised learning. Generalization means that an NN classifies the input data to a desired value, even though input data is not the same as the desired value. Associative memory creates output even when confronting unfamiliar or incomplete data. Fault tolerance allows work to continue even when a part of a neuron is faulty or disconnected.

An NN readjusts the weight values during the learning procedure, so that the NN outputs the closest value to a desired value whenever new data are entered.

On the one hand, the NN is time-consuming during the learning procedure, and needs massive data. On the other hand, once the system is learned, it does not take long for the process to complete recognitions and data detections.

An NN is also used to model complicated problems which are difficult to make equations by analytical methods. For example, an artificial leg can work through the co-operation of sensor groups after each of sensor groups pre-learned about standing, sitting, or running procedures.

In this thesis, data consists of two groups: face and non-face data. Two data groups were used to test the classification ability of an NN.

2-2-2 Learning and Detection Algorithm

Neural networks are divided into several groups (refer to Figure 2-1) [24]. In our research, multi-layered perceptron (MLP) was used as a representative method for the supervised-learning groups. For reference, a single-layered perceptron cannot map a curved line, and cannot solve non-linear problems, like the XOR problem. Therefore, the single-layered perceptron is used for a simple filter or detector due to its limited abilities.

The MLP can classify complex data groups onto a curved line.

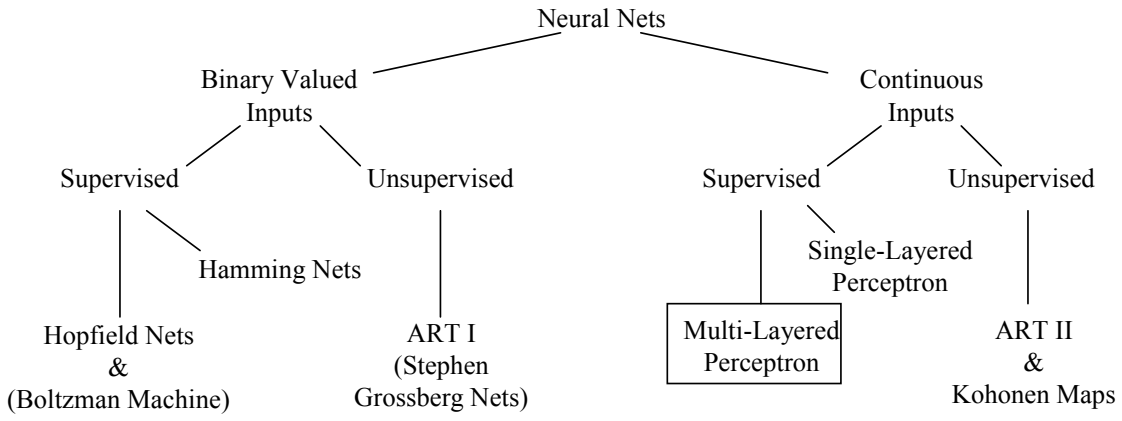
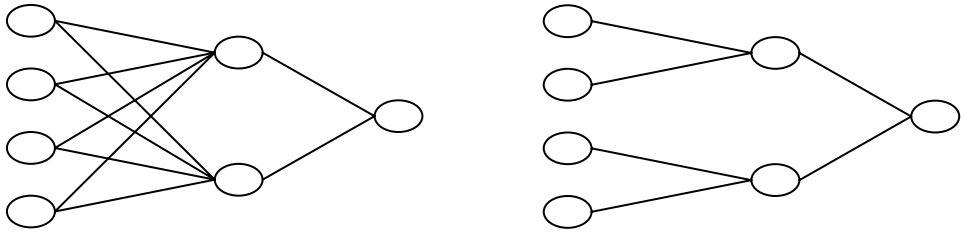


Figure 2-1 Types of neural networks [24]

Considering the topology of an MLP, the MLP can be categorized either as a fully connected, or as a partially connected network as shown in Figure 2-2. A fully connected network is connected by all nodes between layers, unlike a partially connected network which is not connected at all nodes between former and next layers [22]. A partially connected network can be used as a filter. The partially connected network uses less

weights memory; however, it was proved that less connection reduced the performance through the simple experiment.



(a) Fully connected network

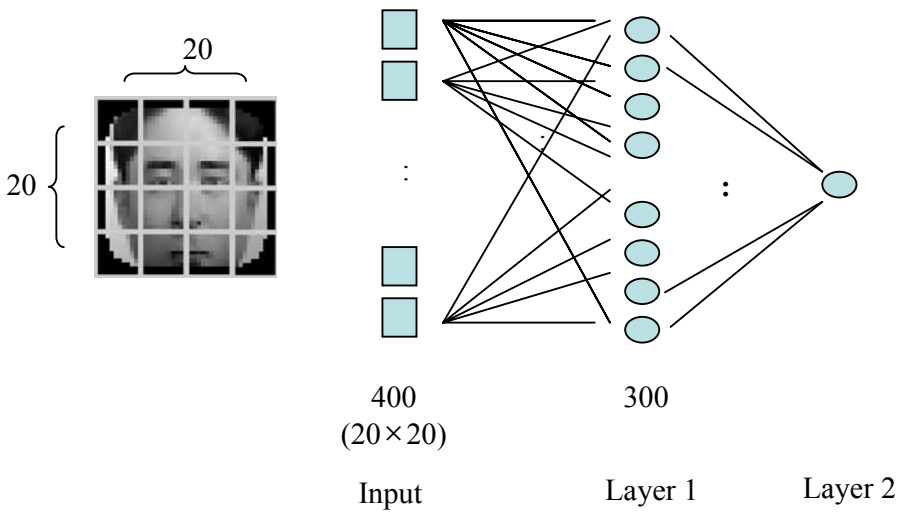
(b) Partially connected network

Figure 2-2 Fully and partially connected network [22]

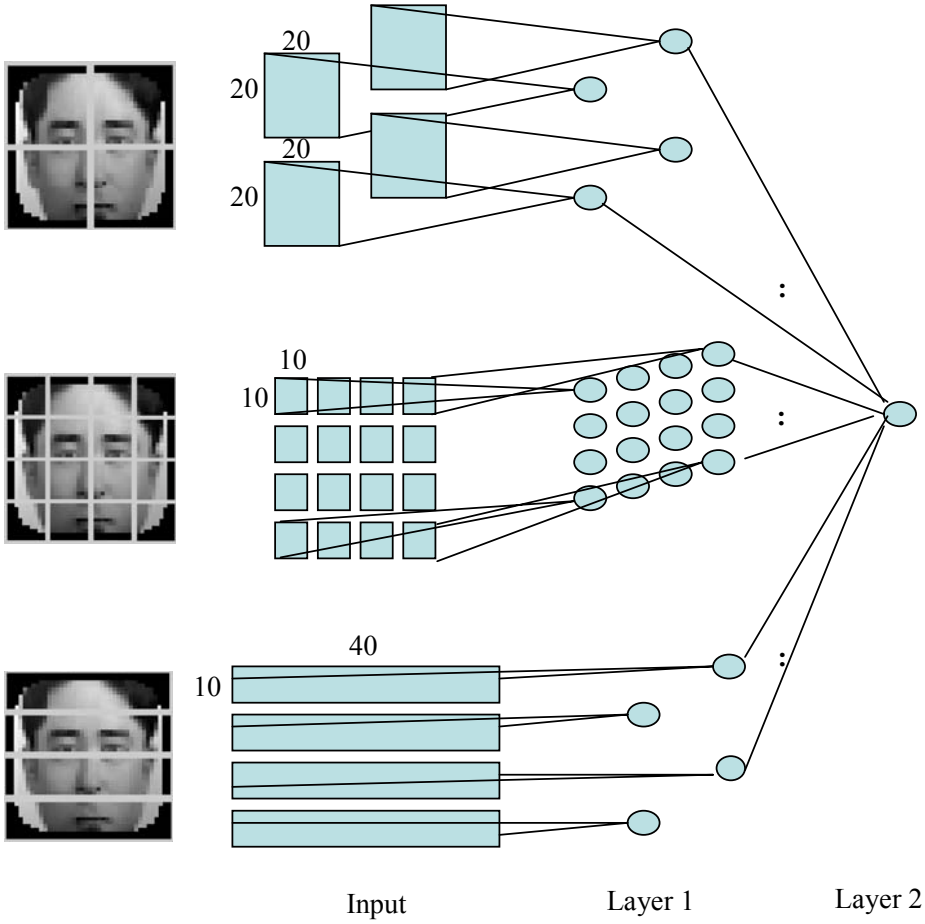
Figure 2-3 shows an example of a fully connected and partially connected network applied for face detection system. This topology divided the part of data to some block, and is known for being efficient when one is detecting the performance [25]. Figure 2-3 reduce the weights by 96%, from (a) to (b), from 120,300 (=400×300+300) to 4,803 (= (20×20×4)+(10×10×16)+(10×40×4)+3).

However, considering its performance, the detection rate of a fully connected network was better than a partially connected network. Moreover, it was very difficult to make the NN learn.

Therefore, multi-layered and fully connected networks were used for the NN detector.



(a) Fully connected network



(b) Partially connected network

Figure 2-3 Example of fully and partially connected network

A learning algorithm of an MLP is called a multi-layer back propagation (MLBP).

The MLBP algorithm renews the weights by reducing the mean squared error (MSE) between the desired value and output of an NN. Equations (2.1) ~ (2.4) show the concept. The MSE of output $E(w)$ with a desired value of (d) and output of (O) can be described in equation (2.1) [22].

$$E(w) = \frac{1}{2}\sum (d - O)^2 \tag{2.1}$$

The error of the weights is the differential of (2.1) and induced to (2.2).

$$\Delta W = \nabla E(w) \tag{2.2}$$

After solving the equation (2.2), the error equation form (2.3) is obtained.

$$\Delta W = (d - O) f'(net) O = \delta O \tag{2.3}$$

Finally, new weights can be renewed by the error equation of weights.

$$W(t+1) = W(t) + \Delta W \tag{2.4}$$

All other extra equations for the learning algorithm are omitted to avoid expansion.

The learning and detection procedure are explained with the notification depicted in Figure 2-4.

$f(x)$ is an activation function which is used to transform the activation level of a neuron to an output [26]. Sigmoid function is used for the activation function due to some of its advantages [27, 28]:

1. Nonlinearity makes the learning powerful.
2. Differential is possible and easy with simple equations.
3. Negative and positive value makes learning fast.

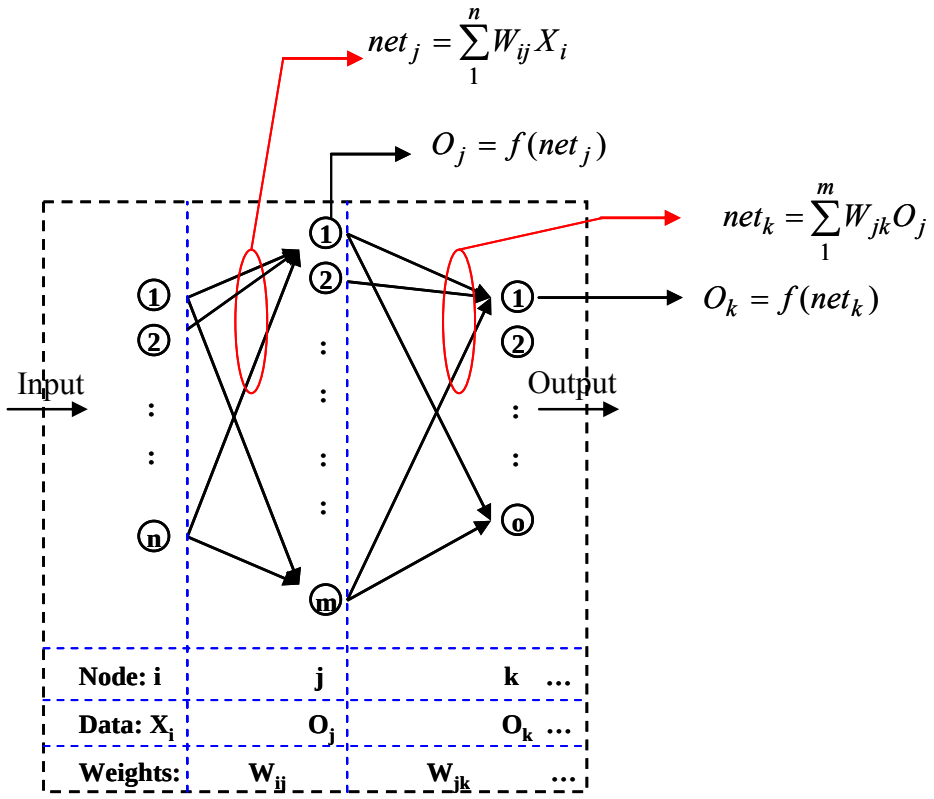


Figure 2-4 A feed-forward neural network (FFNN)

The back propagation learning algorithm is performed in the following steps.

1. Initialization: weights, stopping condition (ex: "if error is less than 0.001"), desired value (ex: Face: 0.9, Non-face: -0.9)

[Feed-forward step]

2. Entering an input pattern.

3. Calculation of input value for j-th neuron (net_j).

$$net_j = \sum_{i=1}^{n} W_{ij} X_i \tag{2.5}$$

4. Output (O_j) calculation of j-th neuron using activation function (sigmoid function).

$$O_j = f(net_j) \tag{2.6}$$

5. Calculation of the input value for k-th neuron (net_k) using the output of hidden layer (O_j).

$$net_k = \sum_{j=1}^{m} W_{jk} O_j \tag{2.7}$$

6. Output (O_k) calculation of k-th neuron.

$$O_k = f(net_k) \tag{2.8}$$

[Back-propagation step]

7. Calculation of the error (δ_k) between desired value and output value

$$\delta_k = (d_k - O_k) f'(net_k) \tag{2.9}$$

8. Calculation of the error (δ_j) of hidden layer

$$\delta_j = f'(net_j) \sum_{j=1}^{m} \delta_k W_{jk} \tag{2.10}$$

9. Renew the weights using the error from (2.9) and (2.10)

$$W_{jk}(t+1) = W_{jk}(t) + \delta_k O_j \tag{2.11}$$

$$W_{ij}(t+1) = W_{ij}(t) + \delta_j X_i \tag{2.12}$$

10. Go to the step (2) and repeat the renewal the weights until the stop condition is satisfied.

After the learning procedure and weights are fixed, the feed-forward step is used for the NN detector, which is called a feed forward neural network (FFNN).

2-2-3 Arithmetic Unit for an Hardware based NN

For the VHDL program to design the FPGA based face detector using the (ML)FFNN, two arithmetic units, multiplication and addition were required as shown in Table 2-1.

Arithmetic is necessary for the following reasons:

1. to add and multiply the input and weights data (or former layer data and weights),
2. to calculate output value using the activation function,
3. to determine face or non-face through the comparison of output and threshold value.

The activation function was estimated by a polynomial equation so that multiplication was used. The determination procedure is obtained from subtraction, and subtraction is calculated from addition. Therefore, all required arithmetic units are (FPU) addition and multiplication.

Table 2-1 Arithmetic units for the FPGA based NN detector

Function	Arithmetic unit
Net value calculation $(net_j = \sum_{i=1}^{n} W_{ij} X_i,\ net_k = \sum_{j=1}^{m} W_{jk} O_j)$	$\times$, $+$
Activation function calculation $(O_j = f(net_j),\ O_k = f(net_k))$	$\times$ (by polynomial estimation)
Classification value calculation $(O_j - threshold)$	$+$ (with two's compliment)

2-3 Floating-Point Unit and Standard

Finite precision number system need to be defined and designed for all hardware or software systems, due to their limited resources. There are two kinds of representative number representation methods: an FXU and an FPU. An FXU has the fixed-point and an

FPU express the point by exponent bits; therefore, an FPU consists of sign, fraction, and exponent bits. The exponent expression makes it possible to express the number having wide range. A general computer uses both an FXU and FPU inside of a CPU because of its advantages: wide range of an FPU and the accuracy of an FXU.

An FPU and its advantage, wide range, are useful for various systems without concern of an exact range, which causes overflow and underflow error.

The Floating-Point Arithmetic method following the IEEE Standard have been used in our general purpose computers since IEEE announced the IEEE Standard 754 for an FPU in 1985 [29]. Figure 2-5 shows the single and double precision of standard FPU format.

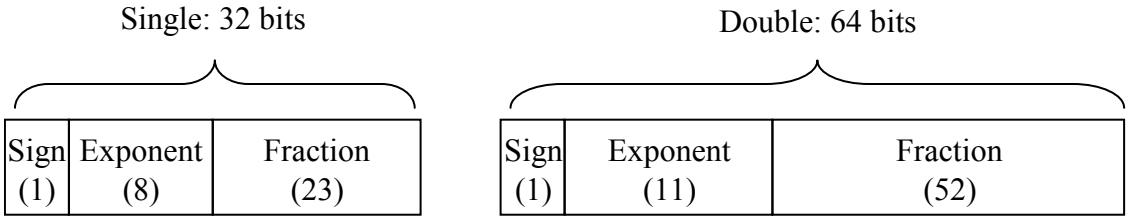


Figure 2-5 FPU standard format (single and double precision)

Representative calculation modules are FPU addition and multiplication because other calculation modules, like division, can be expressed by using both calculations. Therefore, the study of both calculations is important.

Figure 2-6 shows the basic steps for the FPU calculation method which consists of alignment, calculation, normalization, and rounding. At the first step, the fraction number needs to be aligned if the exponents are different. Secondly, the aligned fraction number is added. Thirdly, normalization performs to fit the format and calculates the new exponent. Finally, rounding is performed. This rounding makes a circuit error; therefore,

the circuit error is caused by rounding the added in the NN error model. Chapter 3 explains the rounding error and how affects the NN output.

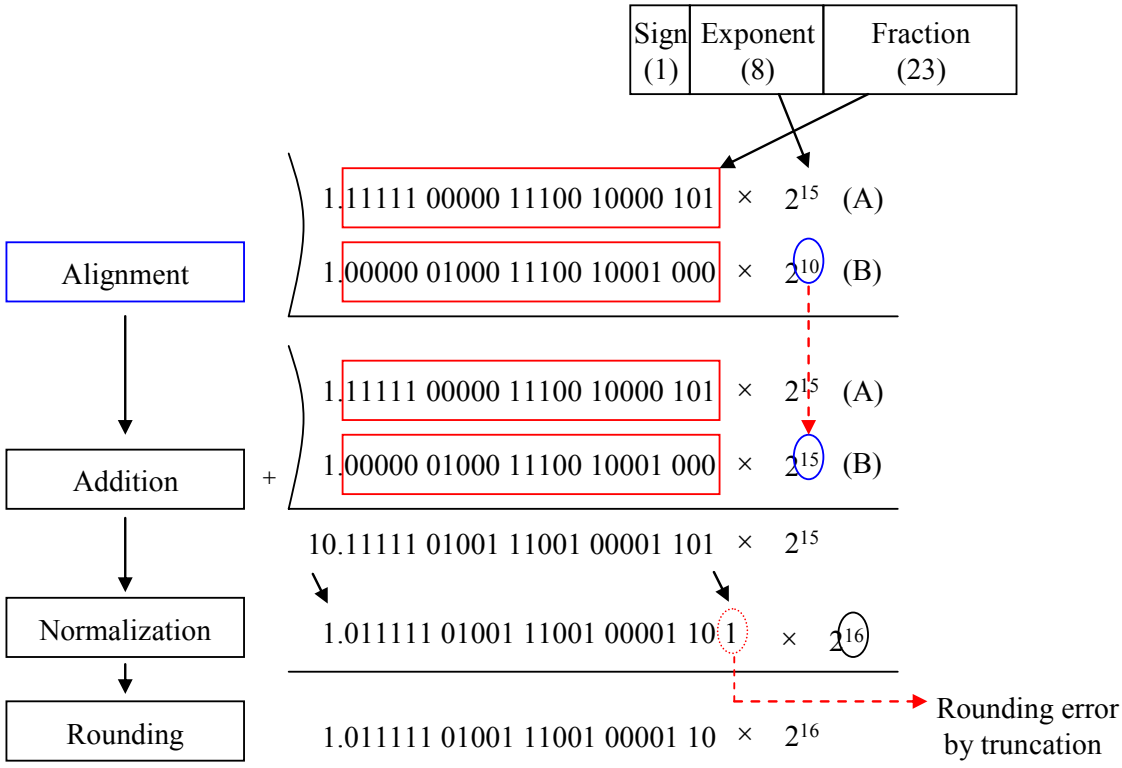


Figure 2-6 FPU addition

In the research, multiplication and addition of the bit-width reduced FPU are designed by following the standard (single precision) using VHDL, and MATLAB was used to verify the circuits. Secondly, the VHDL code was modified to make the circuit adjustable. Then, the experimental errors and synthesis report was obtained.

This adjustable FPU arithmetic IP, design environment to verify circuit and error model was useful to design dedicated embedded hardware system.

2-4 Field Programmable Gate Array (FPGA) Device

2-4-1 Devices for Embedded System

DSP, microprocessor, and FPGA are all possible devices for embedded systems. A DSP is used for signal processing applications like image and sound which require high speed. This high speed feature is performed by pipelining hardware architecture. Other representative advantages of the DSP are fluent libraries for signal processing.

Microprocessors like an ARM, INTEL 80 series, Motorola 68000 series or PIC are the general microprocessor chip for various fields.

An FPGA consists of gates, which are part of the circuit element. An advantage of an FPGA is its flexible digital circuit design, including pipelines and high speeds. Compared with DSP, DSP support 4 pipelines but the FPGA allows the researcher to design a more pipelined architecture. This flexible design feature is more useful to make dedicated system which allows high speed than other devices. Moreover, the flexible design was useful in order to change the hardware architecture.

Therefore, an FPGA is a suitable device for the NN due to its high speed and flexible design.

2-4-2 FPGA Design Environment

To design the FPGA, special programs for the circuit synthesis, porting, and simulation are needed. ALTERA, XILINX, and ACTEL are the most dominant producers in the market, and they provide each of the necessary programs or combined packages of such programs.

Among hardware description languages (HDLs), VHDL and VERILOG HDL are frequently used. For our research, the VHDL was used because it is preferable for parameterization. Figure 2-7 is an emulation program, Integrated Software Environment (ISE) for a XILINX FPGA and show the NN face detector modules and codes.

For a synthesis of logic circuit, the XILINX Synthesis Technology (XST) program inside the ISE was used. The hardware performance of area and timing can be improved by a few percent through the use of a dedicated synthesis tool like SYNPLIFY. After synthesizing the digital circuit, a total gate capacity and maximum operating frequency were obtained. Finally, those calculations allowed us to infer the implementation possibility and features of an NN system design using the FPGA.

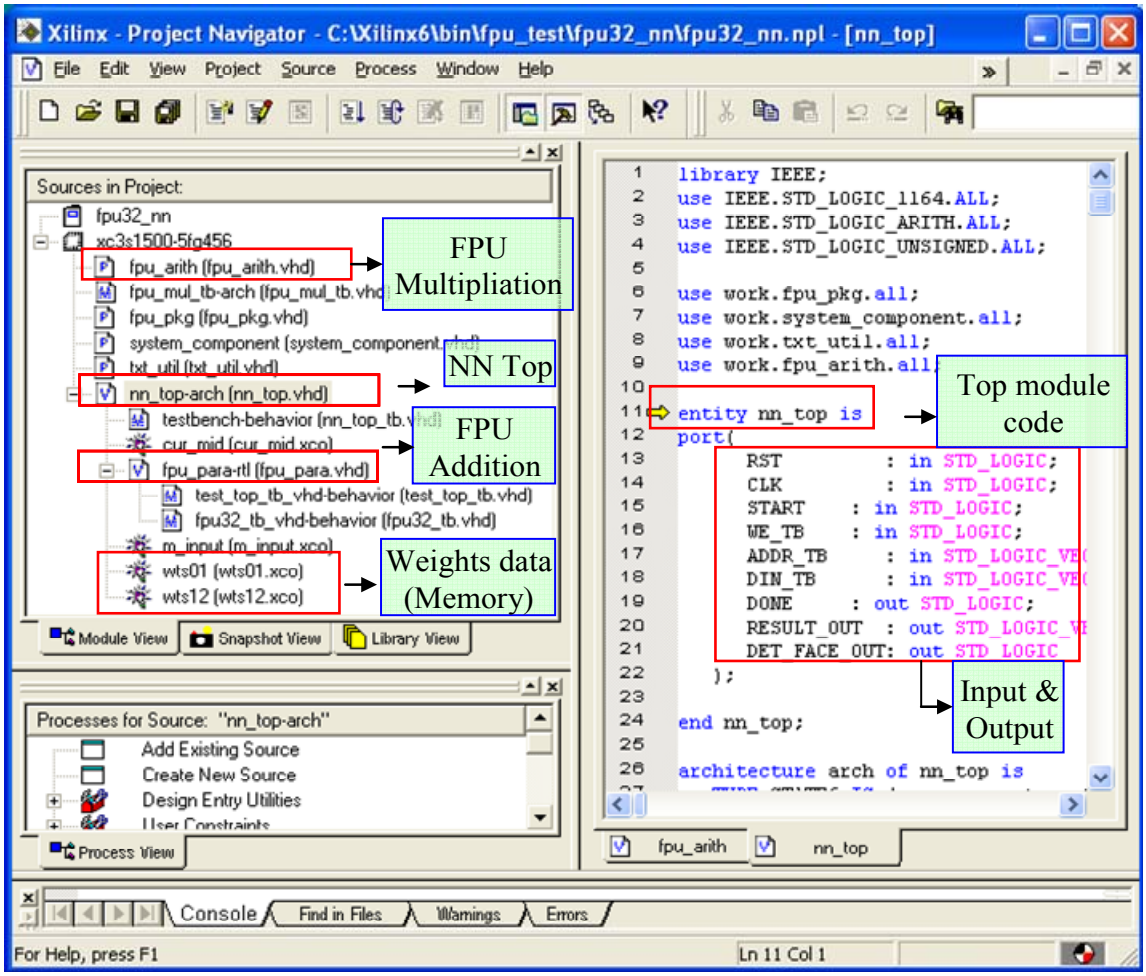


Figure 2-7 XILINX FPGA emulation program (ISE)

In the simulation program market, two programs, a MODELSIM and NCSIM, are used. MODELSIM is used for simulation. Two simulation programs work with a library which is provided by FPGA vendors, ALTERA and XILINX, so that we can estimate the exact delay for the circuits by considering their physical features. Figure 2-8 shows the MODELSIM waveform describing the FPU addition and the multiplication calculations to obtain the net value. To verify the value by watching the all values and waveform is not convenient and is tedious work. For example, a simulation time of 1.914 seconds using 220 files for face and non-face data took one day (and saving few Giga Bytes size

data in a PC hard disk (HDD)). This means it takes one day to check the result after the code is changed. Therefore, an automatic verification using MATLAB had to be developed to help verification.

MATLAB program provides the test-bench file, and reads and analyzes the result file from the VHDL program.

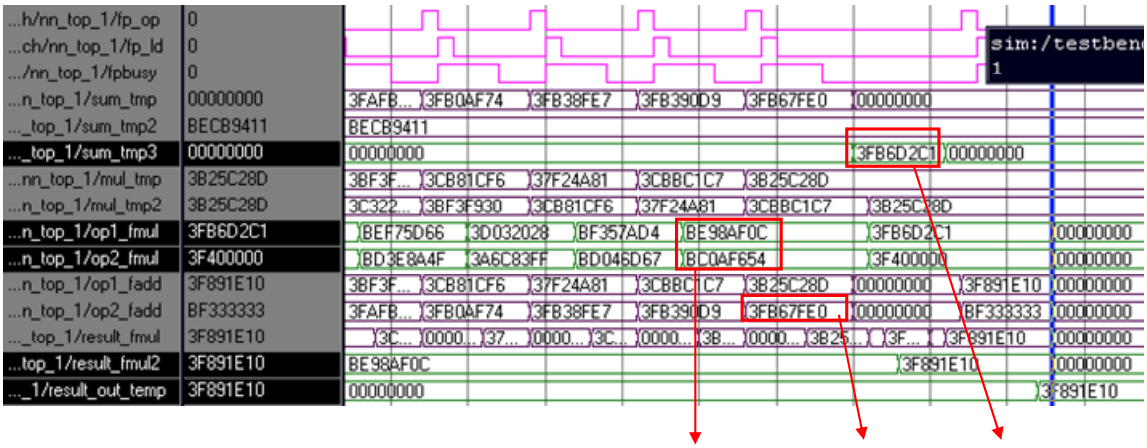


Figure 2-8 Modelsim waveform

XC3S4000 (about 4M Gates), one of the SPARTAN series, was considered as an FPGA device [30]. The chip is not expensive compared with other FPGAs, because it does not have many hardwired IP inside, and it is easily utilized in the fashioning of a simple detection system.

2-4-3 Total Design Flow for the FPGA based NN Detector Design

In the commercial market, some companies provide fixed-point simulation models for application in embedded systems. However, no company provides simulation models based on the bit-width reduced floating-point.

Figure 2-9 shows the design environment for this thesis. The NN face detector was simulated on MATLAB and Visual C++. The MATALB environment simulates the NN face detector, and makes a test-bench for the VHDL program. The MATLAB program also provides the test-bench code for VHDL simulation. The programs create the bit-width reduced FPU format code. The test-bench data consists of face data and weights data.

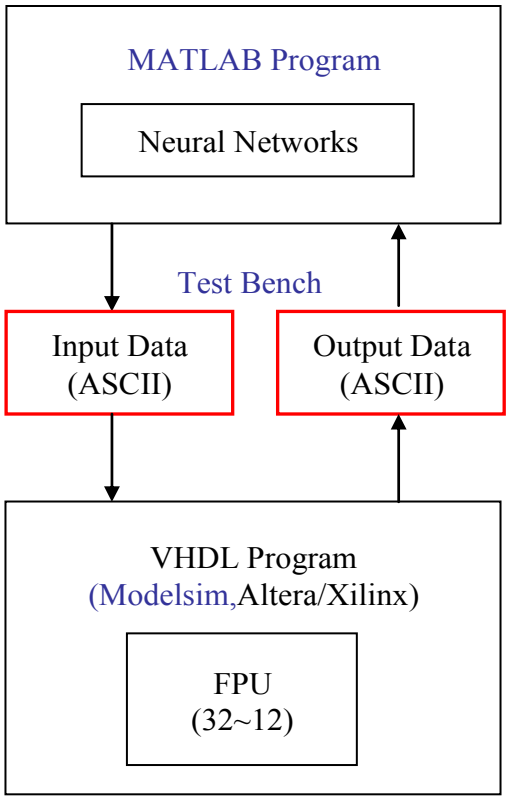


Figure 2-9 Design environments

The Visual C++ can substitute for the MATLAB. The advantages of using Visual C++ is that a Graphic User Interface (GUI) is user-friendly and that it has a fast running time so it doesn't use much of the computer's resources. However, the programming of Visual C++ is not as easy as the MATLAB.

The MODELSIM program supports the VHDL simulation environment. Of the two major HDLs in the market, the VHDL was used because of the VHDL's advantages of commands, package and generic, are suitable for making parameterized circuits. After the MODELSIM program simulates the circuit using the test-bench, it makes the output data. Then, finally, the MATALB programs read the data, and analyze the error and performance. The ISE was used to synthesize the circuit for FPGA. We can develop logic circuits in the FPGA through two tools: simulation and synthesis programs.

The environment, MATLAB+HDL simulator, and the bit-width reduced FPU will be helpful to decide the system specifications for an embedded system.

CHAPTER 3 ERROR ANALYSIS

Unfortunately, there is no computer which can express all numbers due to its finite hardware resources. In practice, a personal computer uses a limited bit width of 64 bits to express the number system. Therefore, any digital system cannot be free from errors. An application, like an embedded system, with a specific purpose needs to be designed and confined enough to satisfy given conditions in order to reduce the unnecessary complexity of components and to save costs. Therefore, it is essential to analyze the error of a system to determine the bits required.

An FPU is useful for an NN in terms of its acceptance of a wide ranging number system and its memory efficiency. The error model with maximum relative representation error (MRRE), which provided representative indices of FPU accuracy, was developed and the total output error of the NN system was analyzed. The main calculation of the NN system is a Multiplication and Accumulation (MAC) process. The FIR filter and the FFT, which are frequently used for engineering, also use the MAC calculation; therefore, this error model can be applied to the FIR filter and the FFT as well.

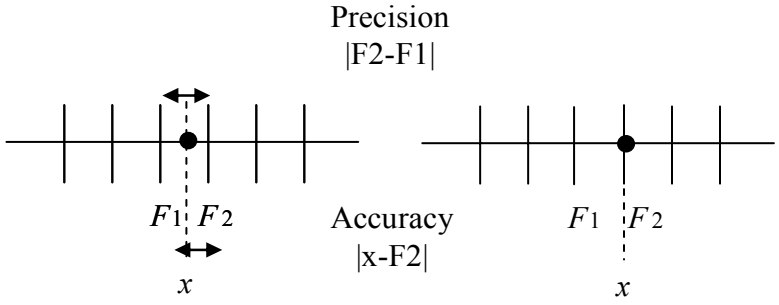
In our design, an NN + bit-width reduced FPU + FPGA, there are two possible errors: polynomial estimation error and reduced precision error of the FPU. Estimation of an activation function causes a polynomial estimation error. In Chapter 5, the side effects of this error are explained by analyzing the experimental result. We are more interested in the second error, Sections 3-1 and 3-2 explain the second error and its side effects.

3-1 Error caused by bit-width reduced FPU

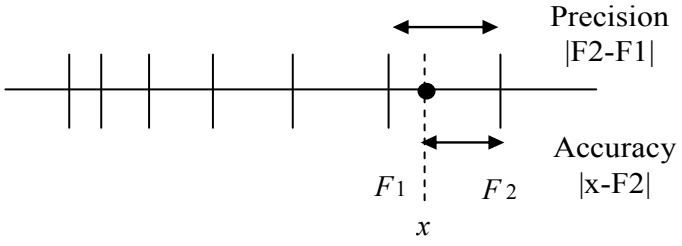
3-1-1 Accuracy and Precision of FPU

Figure 3-1 shows the machine number expression of an FXU and an FPU. The FXU has a fixed distance between consecutive numbers. The FPU have different distance between consecutive numbers. This makes the difference between the FXU and the FPU. The FXU can be more accurate than the FPU in possible range of the FXU; meanwhile, the FPU can express more wide range relatively losing accuracy.

A precision is defined as the distance two consecutive number expression, for example, $|F_2\text{-}F_1|$as shown in Figure 3-1. An accuracy is defined as the distance between the real number (x) and machine number expression, $|x$ -F2|.



(a) Fixed-point number expression



(b) Floating-point number expression

Figure 3-1 FXU and FPU number expression

Figure 3-2 explains the advantage of the wide range of the floating-point arithmetic representation method. The data within specific bounds of the fixed-point method has less errors than the FPU, however, a narrow range make more errors than the FPU when data goes beyond the bounds. The floating-point method, however, maintains an error of the same rate over a much wider range of numbers than the FXU. The advantage of the FPU wide range makes it easy to design when the exact number range is not known and when the data range are changed frequently. This advantage can be a very useful feature for an NN because the range of weights is difficult to know exactly and can be changed following the system. For example, if two NN detectors are under two different conditions like the bright light and the dark light, weights might be changed.

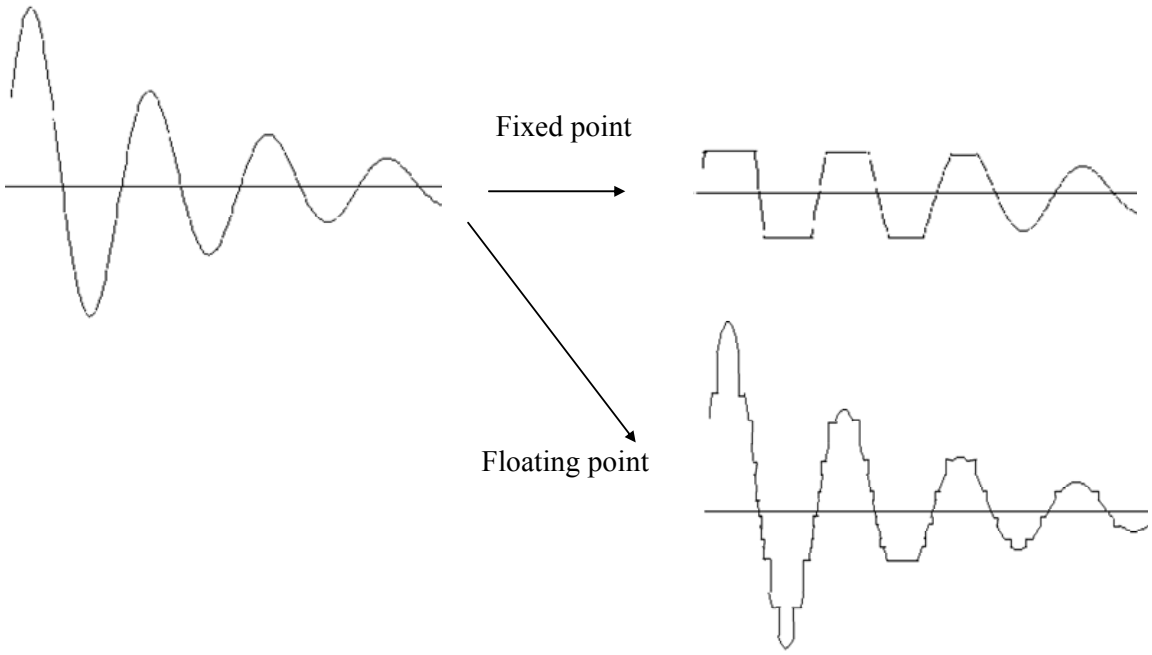


Figure 3-2 Error feature of fixed and floating-point

Figure 3-3 specifically shows how to express a number by the finite FPU. The size of the error can be graphically realized and the error is increased at a constant rate. For example, the error of the value of 8 is less than 0.016, $|8| \times 0.002 \times \leq 0.016$. It means "8" can be expressed by "7.984(8-0.016) ~ 8.016(8+0.016)" in FPU16. Her, "0.002" is called maximum relative representation error (MRRE). Section 3-1-2 explains about the MRRE.

In this example, the FPU16 consist of sign (1bit), exponent (6 bits), and fraction (9bits) respectively.

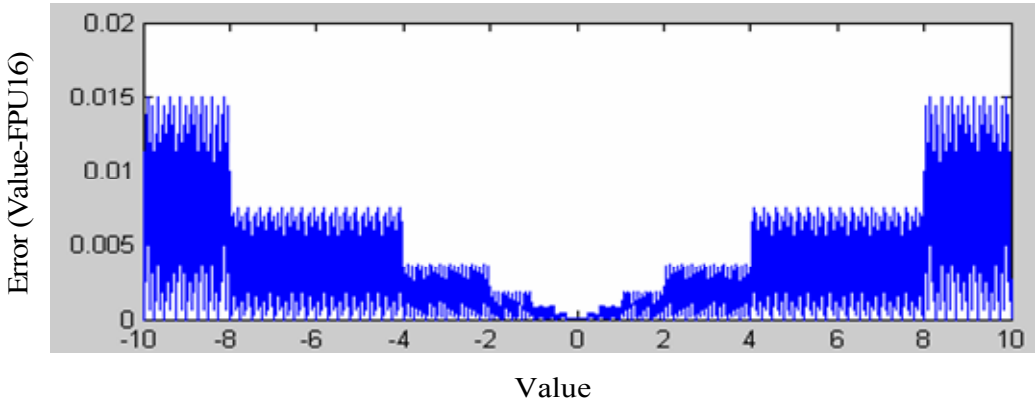


Figure 3-3 FPU16 representation error

3-1-2 MRRE and ARRE

Figure 3-4 shows the relative representation error (RRE) of some FPU16 numbers. MRRE is the maximum relative representation error. The MRRE is the upper bound of the RRE, and is equivalent to the value of unit in the last position (ulp) at radix 2. Equation (3.1) describes MRRE [31].

$$MRRE = 0.5 \times ulp \times \beta \,, \tag{3.1}$$

where β is the radix and ulp is a unit in the last position and ulp is also defined as the distance between two consecutive significands. Therefore, if the fraction bit is 9 (FPU16),the ulp is $2^{-9.}$

For example, let β =2(radix), then the unit in the last position (ulp) is 2^{-9} for FPU16.

$$MRRE = 0.5 \times 0.002(\approx 2^{-9}) \times 2 = 0.002 \tag{3.2}$$

Therefore, the number of the FPU16 is bounded by the number × MRRE (0.002),

Error ≤ |FPU16 number| - |number| = |number| × MRRE.

For example, the FPU16 representation error ≤ 10 × 0.002 = 0.02 (Y-axis of Figure 3-4)

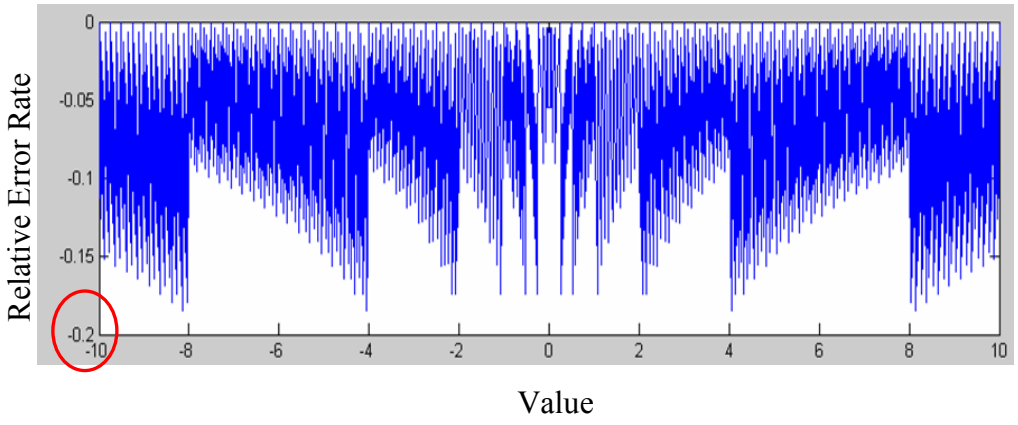


Figure 3-4 RRE of FPU16 (Error/FPU16 number)

An average relative representation error (ARRE) in practical use is shown in equation (3.3). The trend of the graph is not changed by the ARRE because the ARRE is the MRRE of the constant value intervals (Table 3-1). Equation (3.3) describes the ARRE [31].

$$ARRE = \frac{\beta - 1}{\ln \beta} \times \frac{1}{4} \times ulp \,. \tag{3.3}$$

The number of bits in the FPU is important for the area and operating speed [32]. Therefore, it is important to decide on the least number of bits required within the acceptable error range. A maximum relative representation error (MRRE) [31] is used as one of the indices of floating-point arithmetic accuracy, as shown in Table 3-1.

Table 3-1 MRRE and ARRE of five different FPUs

Unit	β, e, m	Range	MRRE (ulp)	ARRE (0.3607×ulp)
FPU32	2, 8, 23	$2^{2^8-1} = 2^{255}$	2^{-23}	0.3607×2^{-23}
FPU24	2, 6, 17	$2^{2^6-1} = 2^{63}$	2^{-17}	0.3607×2^{-17}
FPU20	2, 6, 13		2^{-13}	0.3607×2^{-13}
FPU16	2, 6, 9		2^{-9}	0.3607×2^{-9}
FPU12	2, 6, 5		2^{-5}	0.3607×2^{-5}

3-2 Output Error Estimation of an NN Caused by Bit-Width Reduced FPU

The FPU representation error increases with repetitive multiplication and addition in an NN. The difference in output can be expressed using the following equations with the notification depicted in Figure 3-5.

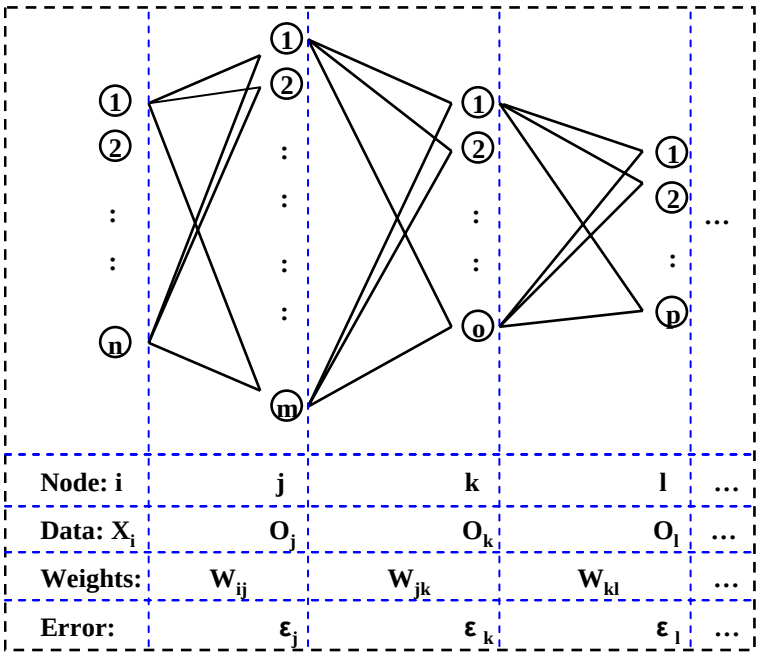


Figure 3-5 A general neural network model

3-2-1 Output Error Estimation by the MRRE and the ARRE

The error of the 1st layer can be described as shown in equation (3.4).

The output error of the first layer, caused by finite length arithmetic unit, ε_j is the differentiation between the output of real number ($O^f{}_j$) and the output of first layer caused by finite length arithmetic unit (O_j), or $O^f{}_j - O_j$. The output is solved by the MAC calculation, $\sum_{i=1}^{n} W^f{}_{ij} X^f{}_i, \sum_{i=1}^{n} W_{ij} X_i$, and the activation function $f(MAC)$ of weights data and input data.

$$\varepsilon_j = O^f{}_j - O_j = f(\sum_{i=1}^{n} W^f{}_{ij} X^f{}_i) - f(\sum_{i=1}^{n} W_{ij} X_i) + \varepsilon_\Phi . \quad (3.4)$$

The error of the 2nd layer also can be described as (3.5).

$$\varepsilon_k = O^f{}_k - O_k = f(\sum_{j=1}^{m} O^f{}_j W^f{}_{jk}) - f(\sum_{j=1}^{m} O_j W_{jk}) + \varepsilon_\Phi \quad (3.5)$$

Where ε_j represents the hidden layer error, ε_k represents total error generated between the hidden layer and output layer. *W* represents the weights and *O* represents the output of the hidden layer [16].

ε_Φ is the summation of other possible errors:

$$\varepsilon_\Phi = \varepsilon_f + \text{Multiplication Error } (\varepsilon_*) + \text{Summation Error } (\varepsilon_+)$$
$$+ \text{other calculation errors.} \quad (3.6)$$

ε_f is the non-linear function error of Taylor estimation. A polynomial equation was used to estimate an activation function so that all derivatives, except the first derivative, are 0. (Equations (4.3) and (4.4), $f''(x)$, $f'''(x)$ …=0). Therefore $\varepsilon_f = 0$.

Other calculation errors occur when the differential of activation is calculated, $f'(x)$=0.75×sum and the final face determination calculation,

$f(face(-), nonface(+)) = sign(f'(x) - threshold)$.

The multiplication unit assigns twice the size of the bits to save the result data. For example, multiplication of 16 bits × 16 bits saved 32 bits in the register. This large size register reduces the error. These errors are negligible except for the summation error (ε_+). The error (ε_+) was added in the error term (ε_Φ). The multiplication error and the addition calculation errors are bounded by the MRRE (assuming rounding mode = truncation) as given by equations (3.7) and (3.8).

$$\varepsilon_* < \text{Multiplication Result} \times (-MRRE). \tag{3.7}$$

(For example, the ε_* of "$4 \times 5 = 20$": $\varepsilon_* = 20 \times (-MRRE)$.)

$$\varepsilon_+ < \text{Addition Result} \times (-MRRE). \tag{3.8}$$

(For example, the ε_+ of "$4 + 5 = 9$": $\varepsilon_+ = 9 \times (-MRRE)$.)

Note that the maximum error caused by truncation is bounded by

$$|\varepsilon_t| \le \left| x \times (2^{-fp} - 2^{-ulp}) \right|. \tag{3.9}$$

In this equation, fp is the final position of the number before being truncated like Table 3-2, and the fp is dependent on the circuit design. This means that the large size registers are assigned and reduced the rounding error. Therefore, (3.9) can be described as,

$$|\varepsilon_t| \le \left| x \times (2^{-fp} - 2^{-ulp}) \right| \approx \left| x \times 2^{-ulp} \right| = \left| x \times (-MRRE) \right|. \tag{3.10}$$

Appendix A-2 explains how to express the bit-width reduced FPU error through the MRRE.

The error by round-to-nearest scheme is bounded as

$$\varepsilon_n \le x \times (2^{-ulp-1}) = x \times \frac{1}{2} \times MRRE \,. \tag{3.11}$$

The total error can be reduced by almost 50% by round-to-nearest scheme.

For example, the max error by truncation, in the case of Table 3-2, is

$$\left|\varepsilon_t\right| \le \left| x \times (2^{-fp} - 2^{-ulp}) \right| = \left| 2^{-2} - 2^{0} \right| = \left| -3/4 \right| .$$

The max error using the round-to-nearest scheme is,

$$\varepsilon_n \le x \times (2^{-ulp-1}) = 2^{-1} = 1/2 \,.$$

Table 3-2 Error caused by rounding: truncation & round-to-nearest

Position		(ulp) *0*	1	(fp) *2*
Truncation	X	0.	1	1
	X by rounding	0.		
	Error (ε_t)	0.	1	1
Round to Nearest	X	0.	1	0
	X by rounding	1.		
	Error (ε_n)	0.	1	0

The error analysis method of the bit-width reduced FPU was summarized as outlined in Figure 3-6, and Figure 3-7 shows the code including input quantization error and rounding error of arithmetic calculation.

1. Define Error Model: $\varepsilon = O^f - O$
2. Substitute Error Term by MRRE:

   $x^f = x \times \varepsilon_x \,,\ \varepsilon_x = x \times (-MRRE)$
3. Add Error Term by Calculation:

   (ε_* = Multiplication Result $\times$ -MRRE),

   ε_+ = Addition Result $\times$ -MRRE.

(a) Summary

1. Function Definition: $O = X_1 + X_2$

2. Error Model Definition

$$\varepsilon = O^f - O = (X_1{}^f + X_2{}^f) - (X_1 + X_2)$$

3. Substitution of Error Term by the MRRE

$$x^f = x \times \varepsilon_x, \varepsilon_x = x \times (-MRRE)$$

$$\varepsilon = \left((X_1 + \varepsilon_{X1}) + (X_2 + \varepsilon_{X2})\right) - (X_1 + X_2)$$

$$\varepsilon = \varepsilon_{X1} + \varepsilon_{X2} = X_1 \times (-MRRE) + X_2 \times (-MRRE)$$

4. Addition of Error Term by Calculation

ε_+ = Addition Result × -MRRE.

$$\varepsilon = X_1 \times (-MRRE) + X_2 \times (-MRRE) + \varepsilon_+$$

5. Final Error Model

$$\varepsilon = X_1 \times (-MRRE) + X_2 \times (-MRRE) + (X_1 + X_2) \times (-MRRE)$$

(b) Simple Example

Figure 3-6 Summarization of error analysis method

```
for k
   for m
      W = LAYER01.Weights(k,m);
      X = NET1_INPUT(k,m);
      SUM=SUM+W*X;
      % Error Term
      h=(2*X*W*MRRE);
      SUM_Error=SUM_Error+h;                    % h
      ADD_ERROR=ADD_ERROR+SUM*MRRE;             % ε+
   end, end
      OUT_LAYER1=0.75*SUM;
      f_diff=0.75;
      Ej(count)= SUM_Error*f_diff + ADD_ERROR;
```

Figure 3-7 Error calculation codes (ε_j)

From the equation (A.23) in Appendix A-3, an error model for an NN with the precision reduced FPU and the MRRE was obtained as equation (3.12).

$$\varepsilon_k < \left| \left(\sum_{j=1}^{m} [(W_{jk} \times -MRRE \times O_j) + (\varepsilon_j \times W_{jk})] \right) \times f'(\sum_{j=1}^{m} O_j W_{jk}) + \varepsilon_+ \right|,$$

$$\varepsilon_+ \approx \sum_{i=1}^{n} (O_j W_{jk}) \times (-MRRE). \quad (3.12)$$

3-2-2 Relationship between the MRRE and the Output Error

From the equation (A.26) in Appendix A-2,

$$\varepsilon_k \approx 2 \times n \times m \times -M \times f'(\sum_{i=1}^{n} W_{ij} X_i) \times f'(\sum_{j=1}^{m} O_j W_{jk}). \quad (3.13)$$

Some properties were derived from (3.13) for the output error. The error is proportional to the number of nodes (n and m) and the MRRE. If the term $n \times m \times -MRRE$ is small enough, the output error caused by the representation error is negligible.

There is one more interesting finding. The differential of summations affects the output error, $\varepsilon_k \propto f'(\sum_{i=1}^{n} W_{ij} X_i)$. Figure 3-8 shows the differential graph of activation function. X-axis shows the output value (= summation value) and Y-axis shows the differential value of activation function. The output of the NN detector which learned well goes to the desired value, 1. In that case, as X-axis value goes to 1, the differential value in Y-axis goes to 0. It means that the well learned NN system has less output error. (Refer to the learning process which makes an NN learn to reduce the error (|desired value - output|).

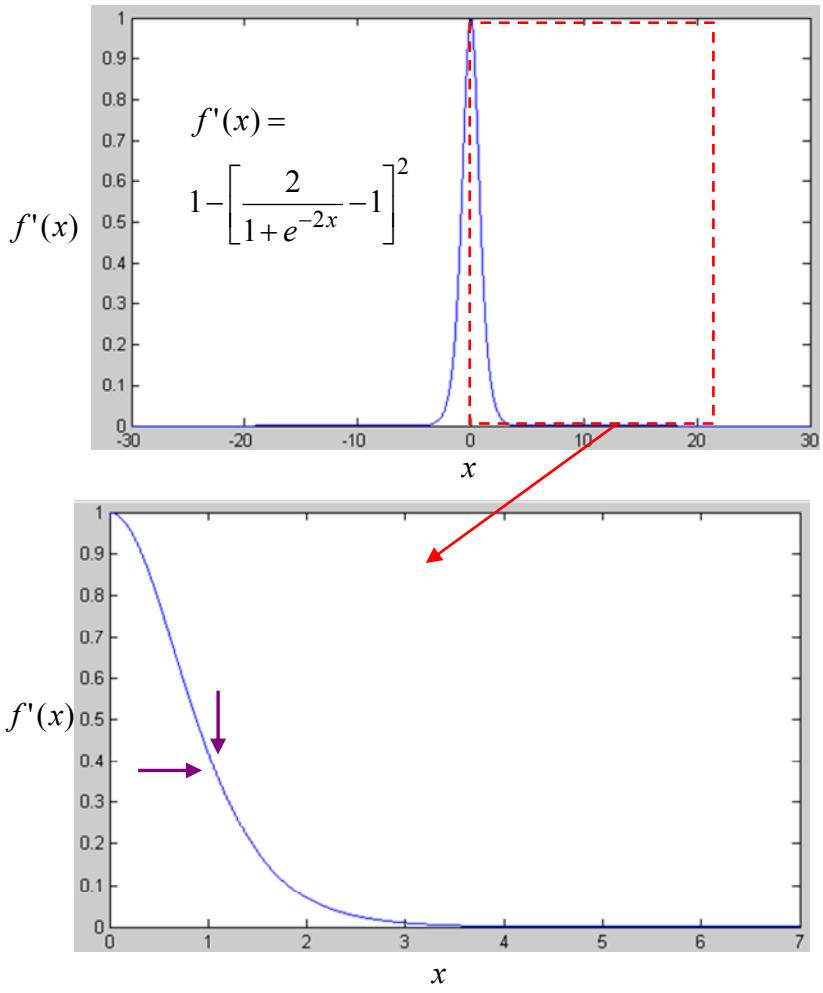


Figure 3-8 First derivative graph of activation function

From the equation (A.22) in Appendix A-1,

$$\varepsilon_j \le \left| 2 \times n \times -M \times f'(\sum_{i=1}^{n} W_{ij} X_i) \right|, \text{ where } M = MRRE \tag{3.14}$$

If W, X < 1,

$$\varepsilon_j \propto MRRE, \tag{3.15}$$

from equation (3.13), the result will be

$$\varepsilon_k \approx 2 \times c \times n \times m \times -M \text{ , c=constant} \tag{3.16}$$

Therefore,

$$\varepsilon_k \propto MRRE\,. \tag{3.17}$$

In our experiment,

ε_k(H300)= 0.002939×400×300×-MRRE=352.68×-MRRE,

ε_k(H500) = 0.002729×400×500×-MRRE=545.8×-MRRE,

H300 and H500 mean 300 and 500 nodes of the first layer respectively. The constant is 352.68 for an NN (H300), and the output error is proportional to the constant and the MRRE. The MRRE is ulp. The ulp value can be expressed by 2^{-n} (n= the number of fraction bits) (refer to eq. (3.1)).

Therefore, the output error logarithmically increases in base 2.

$$\varepsilon_k \propto MRRE = ulp = 2^{-n} \tag{3.18}$$

Finally, it is concluded that n-bits reduction in the FPU creates 2^n times the error. If one bit is reduced, for example, the output error is doubled ($2^{-(-1)}$=2).

3-2-3 The Number of Nodes and Layers

The sensitivity (3.20) can be defined to understand the relationship between the output error and a number of nodes.

$$y = 2 \times c \times n \times m \times M \tag{3.19}$$

$$Sensitivity = \frac{\Delta y}{\Delta M} = 2 \times c \times n \times m \tag{3.20}$$

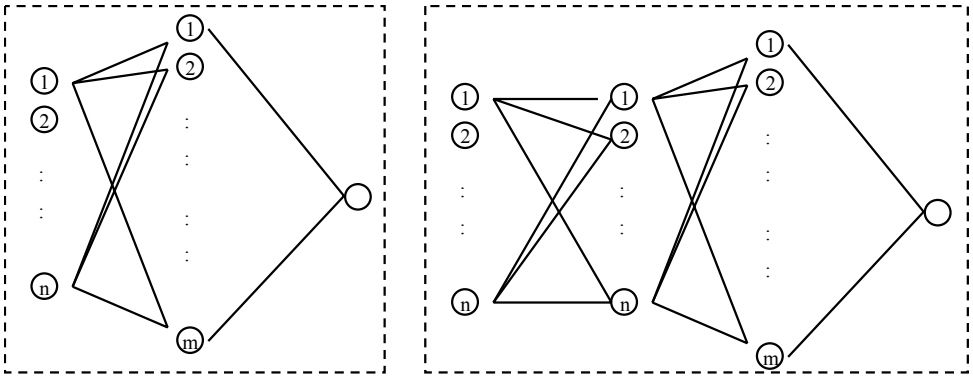
Therefore, if the number of the node is big, the system is unstable, and makes errors proportional to this number of times.

Referring to (3.20), “n” is the number of input (former Layer) and “m” is the number of the first layer (current Layer) node.

Therefore, the number of layers does not affect the error as much as the number of nodes. For example, as indicated by Figure 3-9, the sensitivity of network (a) and network (b) are equivalent.

$$\text{Sensitivity of network (a)} = 2 \times c \times 400 \times 300$$

$$\text{Sensitivity of network (b)} = 2 \times c \times 20 \times 20 \times 300$$



(a) 400-by-300, 2 Layer (b) 20-by-20-by-300, 3 Layer

Figure 3-9 Two networks

3-3 Detection Rate Changes Caused by Output Error

How the detection rate changes by the number representation error can be more important information than the representation error itself. If we know the NN output error caused by the representation error and expected detector performance graph, we can estimate the detection rate changes caused by number representation error.

The NN output error caused by the representation error was calculated through the error model. The expected detector performance graph, Equal Error Rate (EER) graph, can be simulated by Gaussian Distribution Function. Section 3-3-1 explains about EER graph and Section 3-3-2 explains how much detection rate changes by output error

caused by number representation error. This pre-analysis concept will be helpful to estimate the detector performance before designing the embedded system.

3-3-1 A Face Detector Performance Test

Table 3-3 shows the representative statistical measure for a binary classification test [33]. Detector makes four conditions:

1. True positive: Ex. Face data classified to face data.
2. False negative: Ex. Non-face data classified to face data. (Wrong prediction)
3. False positive: Ex. Face data classified to non-face data. (Wrong prediction)
4. True negative: Ex. Non-face data classified to non-face data.

Among them, false acceptance rate (FAR) and false rejection rate (FRR) are used for the representative measure of a face detector performance test. FAR means that face data is classified as non-face data. Therefore, if FAR is 0.01, face can be classified to non-face data. Therefore, the following Table 3-3, FAR = 1 – True Positive.

Negative case and non-face also need to be considered for the total detection rate. FRR = 1-True Negative.

Table 3-3 Statistical measure of a binary classification test [33]

Condition / Test Outcome	True	False
Positive (Face)	True Positive (Face detection rate among Face data base)	False Positive (FAR)
Negative (Non-face)	False Negative (FRR)	True Negative (Non-face detection rate among Non-face data base)

Figure 3-10 shows the EER graph which described FAR and FRR. The Gaussian distribution function (eq. (3.21)), also called normal distribution, was used to estimate the EER graph. The graph is adjustable, changing two variables a mean (μ) and variance (∂).

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right) \tag{3.21}$$

An EER was named for this graph because the left and right regions are the same from the cross point.The X-axis shows the threshold value (to determine face or non-face from the output value of the NN detector) and Y-axis shows the probability. As the EER value is small, the detector is an ideal system because FRR and FAR will be 0. The nearby cross point means that the False Positive (Face detect rate/face data) and False Negative (Non-face detect rate/non-face data) will be 100%.

After the threshold value (output value), changed by the number representation error as entered into x in MATALB program (equation (3.21)), the changes of the detection rate are obtained. Section 3-3-2 describes about the detection error.

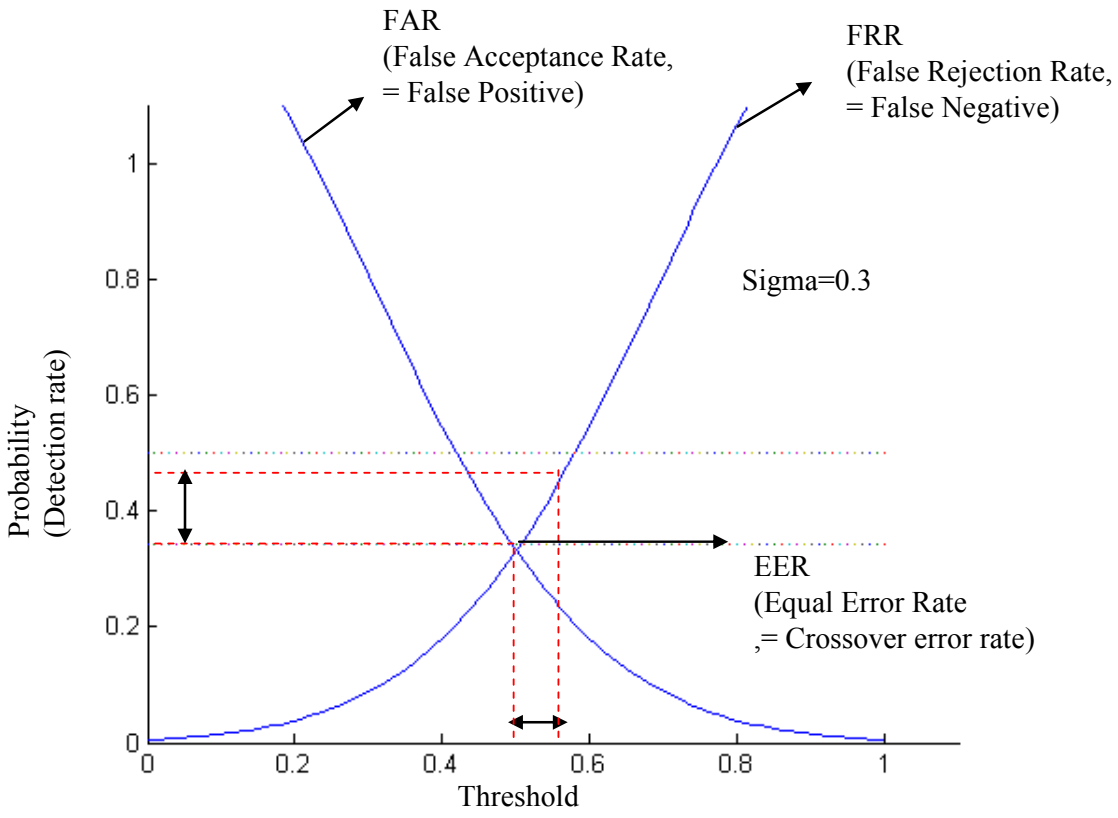


Figure 3-10 EER graph of detection system

3-3-2 Detection Rate Changes Caused by Output Error

In order to approximate the detection rate errors before designing the hardware system using the VHDL, an equal error rate (EER) graph was developed using the Gaussian Distribution Function. The EER graph shows the maximum detection rate error versus the threshold error variation as shown in Figure 3-10.

The bit reduction of 25% from the FPU32 to the FPU24 is affected by the maximum 0.48 % deterioration in the detection rate as shown in Table 3-4. This error pre-simulation is useful for estimating the detection error caused by the arithmetic unit error with an expected EER graph. Chapter 5 explains this experiment result.

Table 3-4 Detection rate errors

Unit	MRRE	NN OUT Error (Max)	Detection Rate Errors (1-FAR)
FPU32	2^{-23}	4E-05	7.38E-05
FPU24	2^{-17}	0.0026	0.0048 (0.48%)
FPU20	2^{-13}	0.0410	0.081
FPU16	2^{-9}	0.6560	0.8301
FPU 12	2^{-5}	10.4	0.3316

CHAPTER 4 IMPLEMENTATION

Figure 4-1 shows the total design flow using MATLAB and VHDL. The MATLAB program consists of two parts: learning and detection programs. After the leaning procedure, weights data are fixed and saved to a file. The weights file is saved to a memory model file for FPGA and VHDL simulation. The MATLAB also provides input test data to the VHDL program and analyzes the result from the result file of the MODELSIM simulation program. Pre-processing includes mask, re-sizing, and normalization.

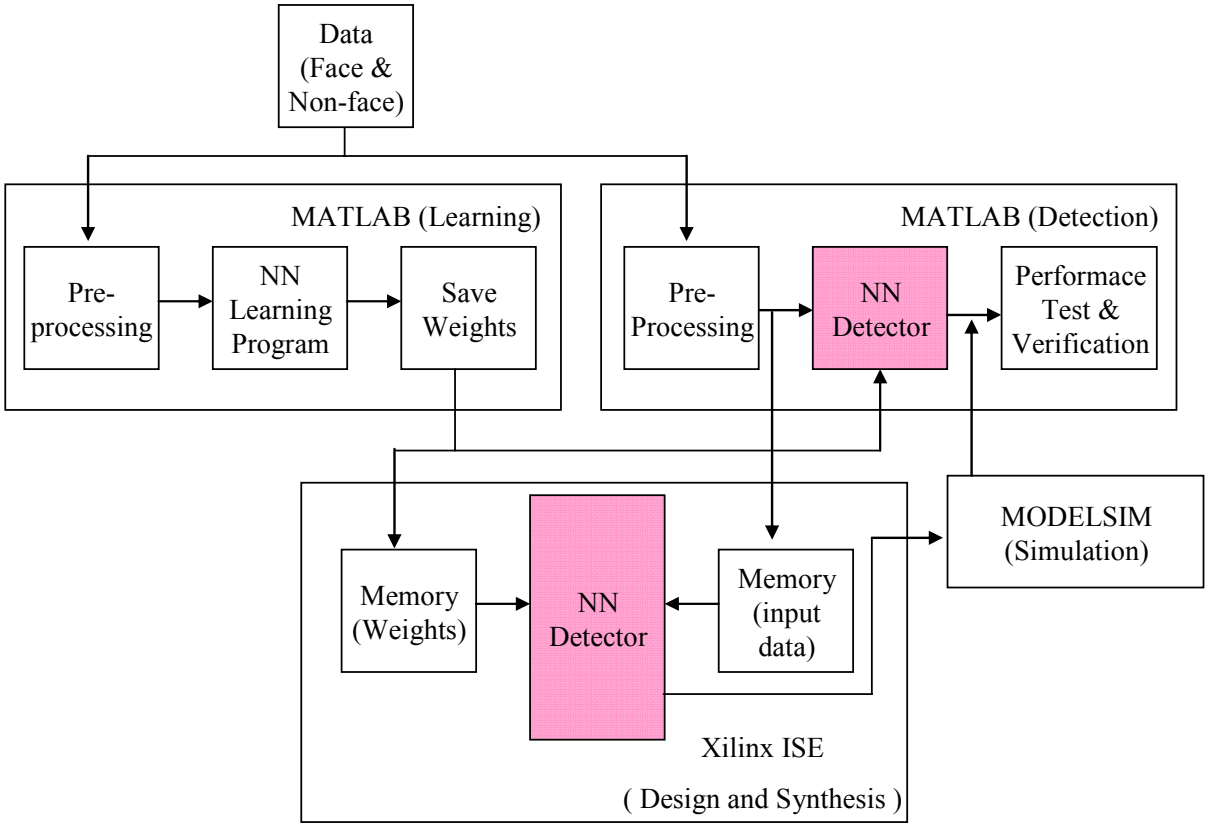


Figure 4-1 Design environment

4-1 An NN Face Detector Design Using MATLAB

In order to get weights data, in other words, to let the circuit learn, the NN used multi-layered back propagation (MLBP) as a learning algorithm.

A multi-layer perceptron (MLP) is a representative method of supervised learning. It is known that 3-layers, having 2-Hidden layers, are better than 2-layers in terms of learning [34].

However, a 2-layer MLP was used in this thesis, as shown in Figure 4-2. The output error equation of the first layer (ε_j : Refer Figure 3-5, eq. (A.3) in Appendix) and the error equation of the second layer (ε_k, eq. (A.7)) are different. However, the error equation of the second (ε_k), the error equation of the third (ε_l, eq. (A-8)) and the error equation of the other layers are the same form. Therefore, a 2-layer MLP is enough to be examined in this thesis.

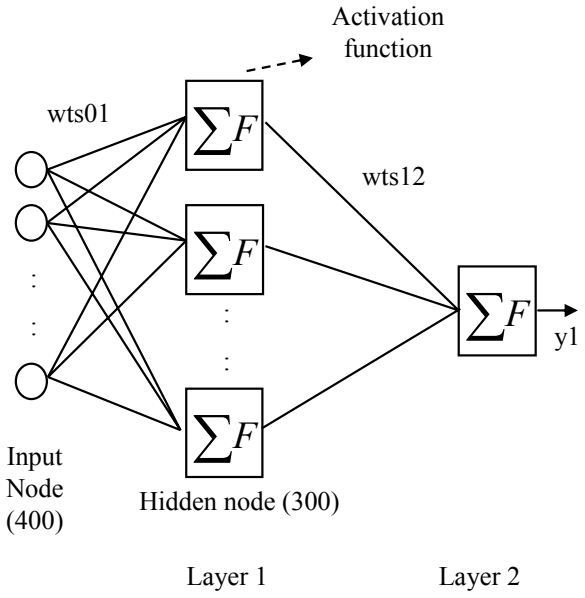
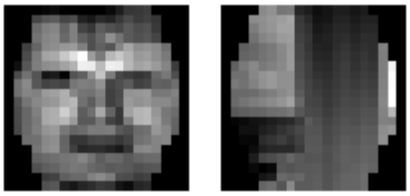


Figure 4-2 The neural network structure (2 Layer Perceptron)

Figure 4-3 shows the sample images of the face and non-face database after the normalization and mask for simple pre-processing. Some institutes and universities provide face databases. The Olivetti face database was used from Scott Sanner's face detector design environment [21] programmed with MATLAB. The Olivetti face data is one of the standard images in the world, and the images consists of mono-color face and non-face image. Some other databases, which have large size, color, mixed with other pictures, are difficult for this error analysis purpose due to the necessity of more pre-processing like cropping, data classification and color model change. Moreover, some databases are not free and require authentication; however, the Olivetti data are free to use.



face data Non-face data

Figure 4-3 Input image (Face and Non-face data)

Figure 4-4 shows the classification result of 220 data including 60 face data and 160 non-face data. X-axis shows the data number of face data from 1 to 60, and non-face data from 61 to 220. Y-axis shows the output value of the NN. The NN was learned to pursue the desired value '0.9' for face and '-0.9' for non-face. Threshold is the standard value of output value to determine face or non-face.

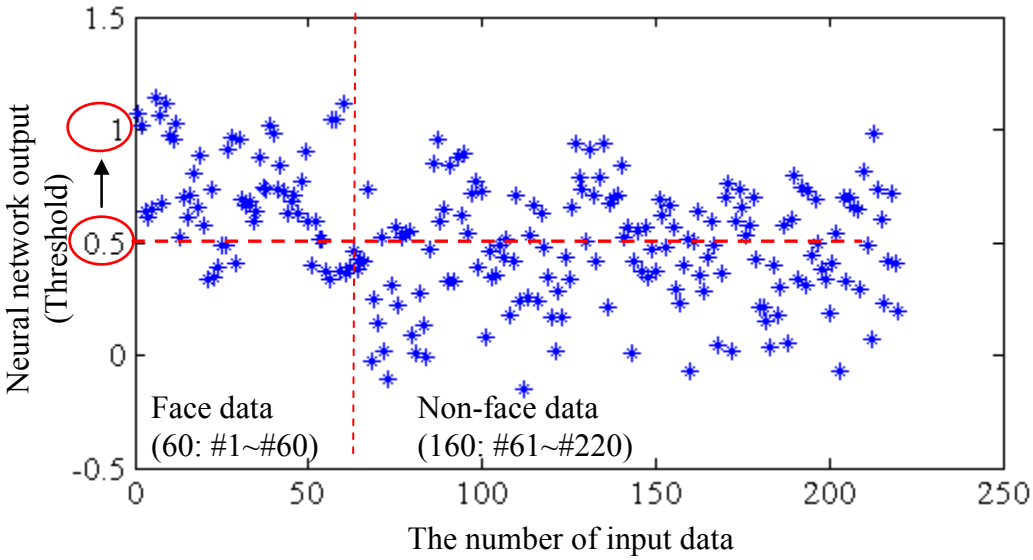


Figure 4-4 Data classification result of the neural network

Table 4-1 was obtained after the threshold value changed from 0.1 to 1. When the threshold is 0.5, the face detection rate is about 83%, and the non-face detect rate is 55%. When the threshold is 0.6, the face and the non-face detection rate are 71.67% and about 69.4% respectively. As the threshold value goes to '1', and as the horizontal red line goes to up in Figure 4-4, the face detection rate is decreased (83%→71.67%). This means the input image is difficult to pass, and it is good for security. Therefore, the threshold is needed to be chosen accordingly depending upon applications. For example, a bank requires more security features than door locks for a car.

Table 4-1 Detection rate of the neural network face detector

Threshold	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Face	60	60	60	53	50	43	29	21	17	10
Rate	100	100	100	88.33	83.33	71.67	48.33	35	28.33	16.67
Nface	17	26	41	65	88	111	130	149	155	160
Rate	10.625	16.25	25.625	40.625	55	69.375	81.25	93.125	96.875	100
Total	35	39.09	45.91	53.64	62.73	70	72.27	77.27	78.18	77.27

The EER graph can be obtained from Table 4-1.

Figure 4-5 is the Equal Error Rate (EER) graph which is one of the best recognition system performance test measures. The EER graph provides a hint of the threshold value decision for specific applications. For example, if the designer changes the threshold from the crossing value, 0.6, toward right side value, 0.7, the FRR is changed from about 30% (0.3) to about 50% (0.5). This means that the face detection rate is decreased from about 70% to about 50%, and the FAR is changed from about 30% to about 20%, so the non-face detection rate was increased to 80%. It means input images, whether the input data is face or non-face, are difficult to pass so that lower face detection rate contributes the security. Meanwhile, as the threshold goes to the left, the input data can be easily passed. The graph provides us a means to select an appropriate balance between security and convenience.

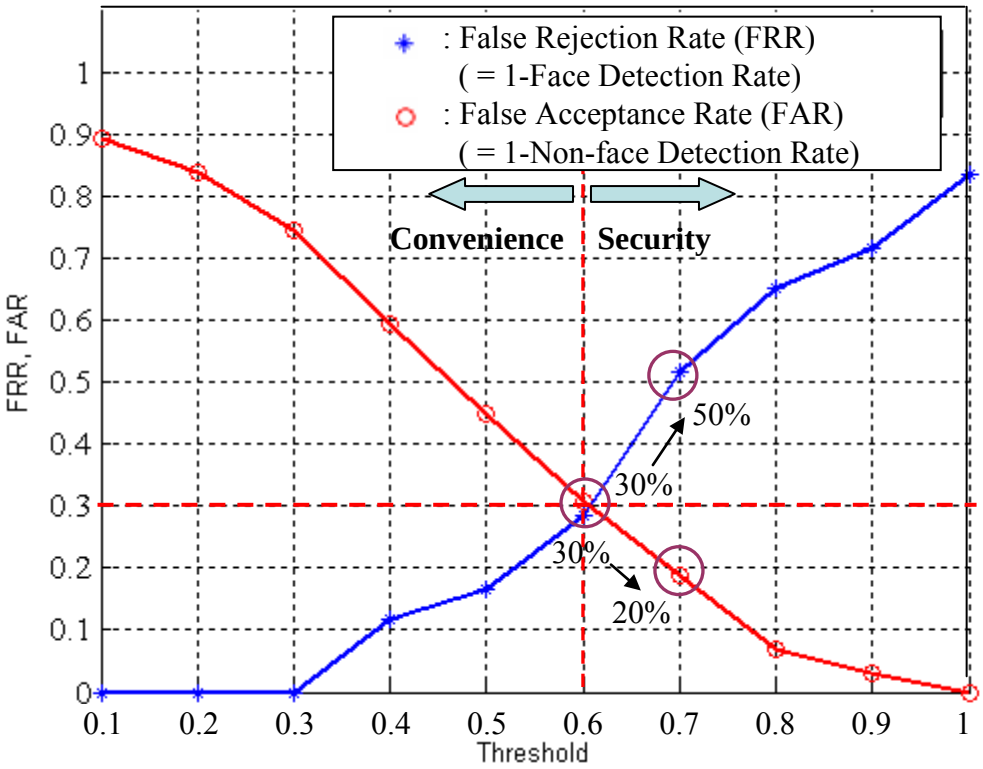


Figure 4-5 Equal Error Rate (EER) Graph

The EER graph was used to estimate how much the number of representation errors affects the detection rate. After designing the VHDL with bit-width reduced FPU, the detection error is obtained directly through the experiment. Chapter 5 and Section 5-2-2 explain a detection rate change by a reduced precision in the FPU.

4-2 An NN Face Detector Design Using VHDL

4-2-1 A Neural Network Design

The implemented neural network architecture uses the multi-layer perceptron (MLP), as shown in Figure 4-6. The architecture is a representative method of supervised learning,

and consists of input and one hidden layer. Each layer has 400 (=20-by-20 images) input nodes for first layer and 300 nodes for second layer, respectively.

The large size nodes (300) of a hidden layer were considered to the worst case because they have the possible biggest error. Note that the number 300 is the largest size to save the weights to a built-in memory of the FPGA (3S4000) at the FPU16 [30].

The size of weights data = 400×300+300×1=120,300=481,200 Bytes

The two layers, hidden and output, share the one multiply-accumulate calculation (MAC) and the activation modules, as shown in Figure 4-6, because the first layer spends 99.8% of the total time allotted to calculate the output. (99.8% = (300×400) / (300×400+300)). Therefore, pipelining to save time was not required.

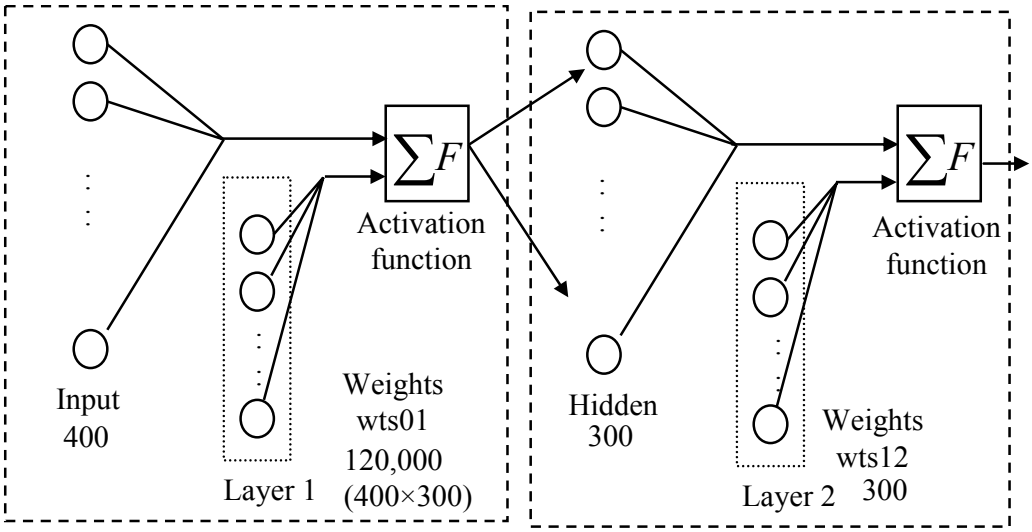


Figure 4-6 FPGA-based MLP structure

After the face data enters the input node, it is calculated by the MAC with weights. Face or non-face data is determined by comparing output results with the thresholds. For example, if the output is larger than the threshold, it is considered as a face data. Here, on the FPGA, this decision is easily made by checking a sign bit after subtracting the output results and the threshold.

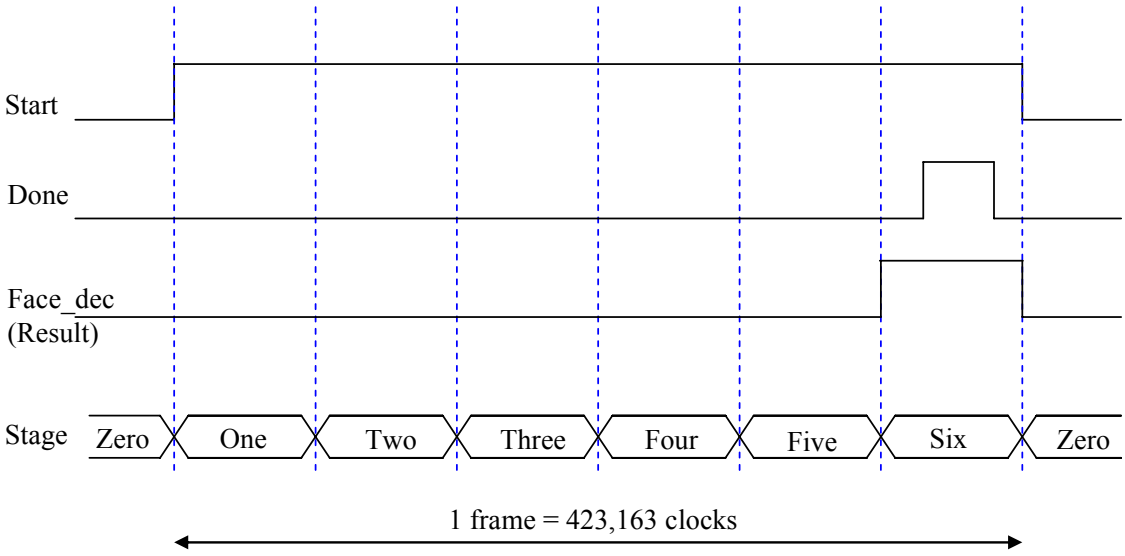
Table 4-2 and Figure 4-7 shows the NN calculation process of one frame in VHDL and final symbol.

Table 4-2 The NN calculation process

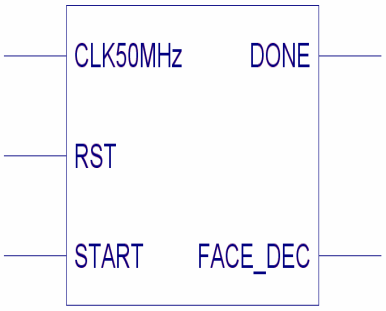
Stage	Description	Operation
One	1st Layer Calculation (MAC of Weights and Input)	$net_j = \sum_{i=1}^{n} W_{ij} X_i$
Two	1st Layer Calculation (Activation Function)	$f(net_j) = 0.75 \times net_j$
Three	2nd Layer Calculation (MAC of Weights and Input)	$net_k = \sum_{j=1}^{m} W_{jk} O_j$
Four	2nd Layer Calculation (Activation Function)	$f(net_k) = 0.75 \times net_k$
Five	Decision of Face or Non-face (If MSB=1, non-face, otherwise face)	$Out(face) = O_k - threshold$
Six	Result Out	

One frame took 423,163 clocks. To process 220 files under the conditions, FPU32 and 48.4MHz, a simulation time unit of 1.914 seconds in waveform was required. It took one day to run the simulation using MODELSIM program.

1.914 seconds = 8.7ms (423,163clocks$\times \frac{1}{48.4MHz}$)$\times$220files



(a) The NN simple waveform and flow



(b) Symbol

Figure 4-7 The FPGA based NN waveform and chip symbol

4-2-2 Activation Function Estimation of a Neural Network

An activation function is used to calculate the output value of the neural network. When learning the procedure, the differentiation of the activation function is used to renew the weights value.

Therefore, the activation function has to be differentiable. A sigmoid function, having an “S” shape, is used for the activation function and a logistic or a hyperbolic

tangent function is commonly used as the sigmoid function. The hyperbolic tangent function and its anti-symmetric feature were better than a logistic function for testing learning ability in our experiment.

For the activation function, the hyperbolic tangent sigmoid transfer function was used, as shown in equation (4.1). The first derivative of the hyperbolic tangent sigmoid transfer function can be obtained as equation (4.2) [23]. The MATLAB provides the commands, "tansig" and "dtansig", for the hyperbolic tangent sigmoid transfer function and its differentiation function.

MATLAB function for (4.1) is "tansig".

$$f(x) = \frac{2}{1+e^{-2x}} - 1 = \frac{1-e^{-2x}}{1+e^{-2x}}. \tag{4.1}$$

MATLAB function for (4-2) is "dtansig".

$$f'(x) = 1 - f(x)^2 = 1 - \left[\frac{2}{1+e^{-2x}} - 1\right]^2. \tag{4.2}$$

The activation function is a transcendental function which is difficult to design in a hardware system. Therefore, the activation function is estimated with all possible hardware resources. (Note that all calculation systems, including computers, estimate the function through a basic arithmetic unit.)

The activation function can be estimated by using multiple methods. A Look-up-table (LUT) is simple and fast, but much more memory is required to make the system accurate; therefore, an LUT is used for a simple NN detector [35, 36]. A Coordinate rotation digital computer (CORDIC) consists of many iterations and takes a long time to calculate; therefore, this algorithm is used for calculators as it does not critically require speed [37]. The Taylor and polynomial methods are effective, and guarantee the highest

speed and accuracy among these methods. The polynomial method was used to estimate the activation function because it is simpler than the Taylor approximation (refer to polynomial estimation equation (4.3) and Taylor estimation equation (B.32)).

Therefore, the polynomial method was applied for our detection system as seen in equations (4.3) and (4.4).

A first degree polynomial estimation of the activation function is

$$f(x) = 0.75 \times x . \tag{4.3}$$

A first derivative is $f'(x) = 0.75 .$ (4.4)

Figure 4-8 shows the activation function and the estimation graph. X-axis is the input of activation function and Y-axis is the output. The activation function confines the output value from 1 to -1. The polynomial function, $f(x) = 0.75 \times x$, can be used for the detector because the output value will be put from the value from -1 to 1 after the learning procedure. Therefore, the region from -1 to 1 polynomial graph in X-axis was used for the NN detector.

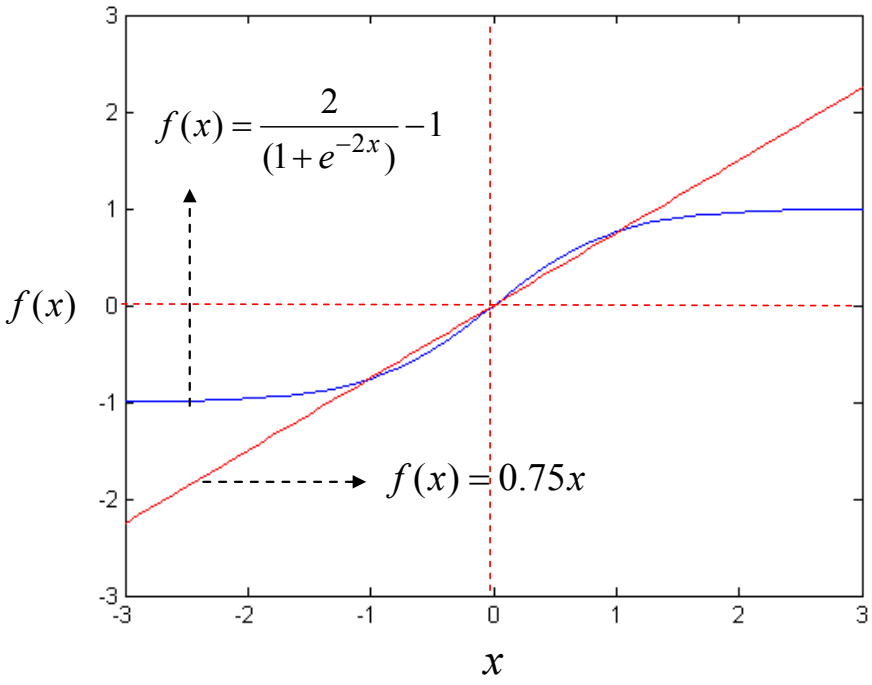


Figure 4-8 Estimation (eq. 4-3) of activation function (eq. 4-1)

4-2-3 Bit-Width Reduced FPU Design

Figure 4-9 shows the NN block diagram in an FPGA. The module consists of control logic and an FPU.

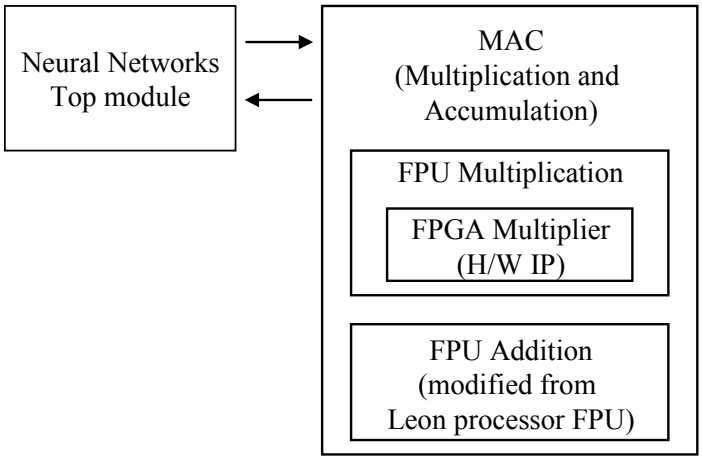


Figure 4-9 Block diagram of the neural network in an FPGA

The implemented FPU is IEEE 754 Standard compliant. In order to make the bit-width reduced FPU simple and to easily to analyze, the number of exponent bits had to be fixed. For example, the FPU64 and the FPU32 follow the IEEE Standard. From the FPU24 to the FPU12, all FPUs have 6 bit exponents. To make the FPU bit adjustable, every sub-module needs to be designed to be adjustable.

The FPU in this system has two modules: Multiplication and Addition. A bit-width reduced floating-point multiplication unit was designed using a multiplier, a Hard IP core in an FPGA, to improve speed.

In our research, the FPU (FPU32) also follows the IEEE 754 Standard single precision (32bits) which consists of sign (1bit), exponent (8bits) and fraction (23 bits). The FPU unit in this thesis consists of two calculation modules: multiplication and addition. A Hard IP core in an FPGA was used to multiply the FPU multiplication module in order to speed it up. The XILINX XC3S4000 [30] is more reasonable in terms of price than a VIRTEX series which has various hardwired IP.

A commercial IP, LEON Processor FPU adder, was used and modified to make the bit size adjustable [38]. The IP was already verified in the market; therefore, design time could be saved and the circuit has proved to be reliable. Figure 4-10 shows the block diagram of the pipelined FPU implemented in this thesis. All modules, multiplication and addition, work in parallel to save time.

The FPU multiplication used the multiplier IP in the FPGA which is a hardwired IP; therefore, multiplication is faster than the FPU addition, because it only takes two clocks. In general, by increasing the number of pipelines, the speeds became faster, but area and power are also increased [32].

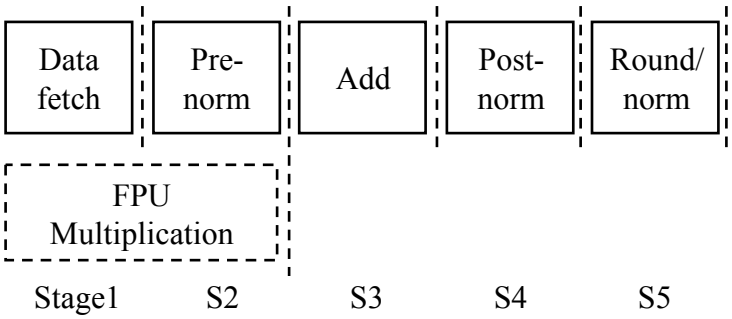


Figure 4-10 Block diagram of pipelined FPU

CHAPTER 5 EXPERIMENTAL RESULTS

5-1 Hardware Performance

The FPGA-based face detector, using the NN and the reduced precision FPUs, was implemented. The logic circuits of the NN face detector were synthesized using the FPGA design tool, XILINX ISE 6.3i on a XILINX FPGA device, SPARTAN-3 XC3S4000 [30]. To verify the error model, first of all, the NN on a PC was designed using the MATLAB. Next, the weights and test-bench data were saved as a file to verify the VHDL code. After simulation, area and operating speed were obtained by synthesizing the logic circuits. The FPU uses the same calculation method, floating-point arithmetic, as the PC so it is easy to verify and easy to change the NN's structure.

5-1-1 Timing

The implemented FPGA detector (the FPU16) took 5.3 ms to process 1 frame, file loading time from memory + NN out time, at 80 MHz which is 9 times faster than 50 ms (40 ms for loading time + 10 ms for calculation time) required for a PC (Pentium 4, 1.4 GHz, like Figure 5-1-(a) as shown in Table 5-1.

Table 5-1 Timing results of the NN and FPUs

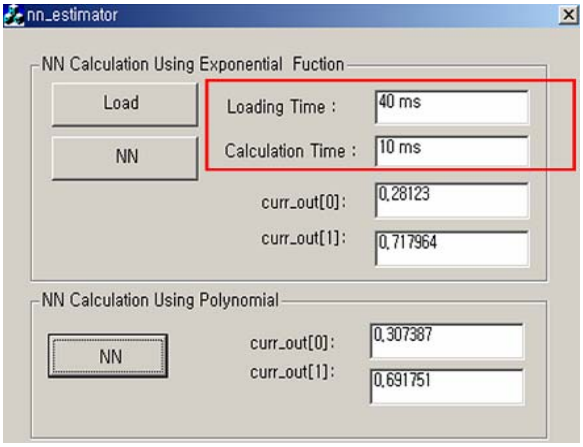
	Max Clock (MHz)	1/f (ns)	Time/1 frame (ms)	Frame rate (frame/sec)
FPU32	48.4	20.7	8.7	114.4
FPU24	57.5	17.4	7.4	135.9
FPU20	77.1	13	5.5	182.1
FPU16	80.4	12.5	*5.3	**189.8
FPU12	85.4	11.7	5	201.8

* NN detect / 1frame =12.5ns× 423,163(total cycle) = 5.3ms

** Frame rate = 1000/5.3=189.8/sec

Figure 5-1-(b) shows how the C program code is used to check how long the program takes. The "GetTickCount" function in Visual C++ returns the number of milliseconds that have elapsed since the program was started [39]. Whenever the program runs, the calculation time is different depending on the PC conditions. 40 ms was the minimum, and normally any function on the program takes at least 10ms, and sometimes the processing time is very slow when the OS is busy.

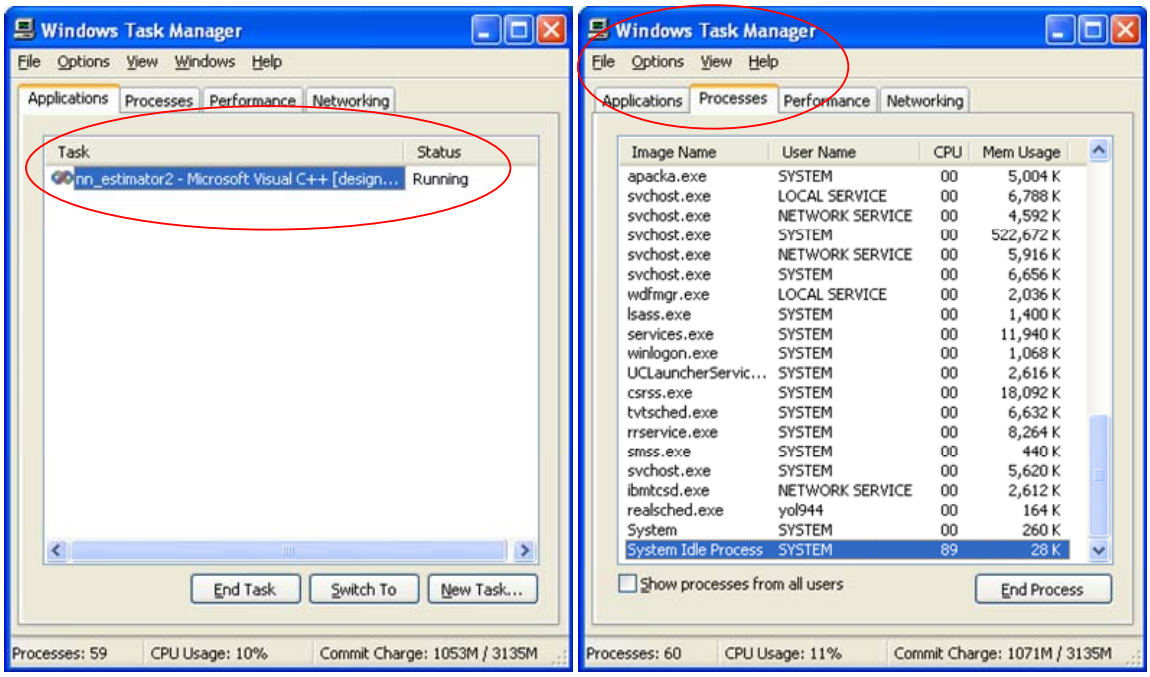
Loading time reads the weights data from the file. The main reason for a long processing time is Windows OS scheduling and the commutation time between hardware resources like hard disk, memory, and the CPU. When the detector is connected to an outer sensor, like a camera or interface system like USB or LAN, processing time will increase. Moreover, other running programs on the OS, like Figure 5-1-(c), will require more time. Therefore, it is obvious that the hardware based dedicated embedded system reduces the processing time and will be very helpful for image processing required high speed operation.



(a) Visual C++ Program

```
DWORD dwFir, dwSec;

dwFir = GetTickCount();                          // t1

:

for(y=0;y<NUM_MID1;y++)

{ wts_01[y]=new double [NUM_IN+1];}

//load_wts_01

FILE *fp1;

if((fp1=fopen(strWeightName,"r"))==NULL)

AfxMessageBox("can't open txt file");

for (k=0;k<NUM_MID1;k++)

for (i=0; i<=NUM_IN;i++)

fscanf(fp1,"%lf",&wts_01[k][i]);

fclose(fp1);

:

dwSec = GetTickCount();                          //t2

m_LoadingTimer.Format("%d ms", dwSec - dwFir);     // Processing time = t2-t1
```

(b) Visual C++ 6.0 Program Code



(c) Windows Scheduling

Figure 5-1 Calculation time in PC

Maximum clock frequencies in the FPGA were 48 and 80 MHz for the FPU 32 and the FPU 16, respectively, which allows enough time to process many frames. For example, the FPU16 provided 190 frames per second. Speed is always critical for image processing because a reduced processing time allows for more image enhancement processing.

Note that the NN, in our research, has large-sized nodes in order to make the larger size error in order to easily analyze the error. If the error is too small, close to 0, it will be difficult to analyze the error.

The FPU arithmetic bit reduction led to a total FPGA based NN area reduction. Bit reduction of the FPU led to an area reduction and a faster operating clock speed. For example, as shown in Table 5-2, the maximum operating clock speed is increased by 61%

(48.4/80MHz). In the case of area, reduction from FPU 32 to FPU 16 resulted in 50% reduction of the adder area (250/486) and in the memory reduction as shown in Tables 5-3 and 5-5.

Table 5-2 Area results of the NN and FPUs

	SLICES	F/F	LUT	MAX FREQ (MHz)
Resource (XILINX 3s4000)	27648	55296	55296	
FPU32	1077(4%)	771(1%)	1952(4%)	48.402
FPU24	878(3%)	637(1%)	1577(3%)	57.509
FPU20	750(3%)	569(1%)	1356(2%)	77.045
FPU16	650(2%)	501(1%)	1167(2%)	80.337
FPU12	556(2%)	433(1%)	998(2%)	85.4

Table 5-3 shows the synthesis result of the FPU adder. A hardwired IP, multiplier block, was used for FPU multiplication, so the adder module is responsible for most of the area in the MAC. A 50% bit reduction from the FPU 32 to the FPU 16 led a reduction by 50% of the adder area (250/486).

Table 5-3 Synthesis result of FPU adder

	Area				Path delay
	Slice	F/F	LUT	IOB	(Max Clock)
FPU32	486	269	905	114	14n (72MHz)
FPU24	403 (83% vs. FPU32)	213	755	90	13n (80MHz)
FPU20	300 (62% vs. FPU32)	185	554	78	12n (88MHz)
FPU16	250 (51% vs. FPU32)	157	461	66	11n (92MHz)
FPU12	173 (36% vs. FPU32)	129	311	54	9.5n (105MHz)

Table 5-4 shows the area of the FPU and the NN.

The area of the FPU adder is 39% (250/650) ~ 45% (486/1077) of the total NN area of the NN as shown in Table 5-4. Therefore, arithmetic size and maximum clock speed are important factors for the system. Note that "%" in FPU adder area represents a percentage of the total size of the NN.

Table 5-4 Area of NN total area and adder (Slices)

	NN total	Adder
FPU32	1077	486 (45%)
FPU24	878	403 (45%)
FPU20	750	300 (40%)
FPU16	650	250 (39%)
FPU12	556	173 (31%)

Table 5-5 shows the memory (weights) size. This result explains why the FPU is important for an NN. The weight memory of the FPU32 is 3,760Kbits (470Kbytes), which is a large size for an embedded system. Bit reduction of the FPU makes it possible

to reduce memory size proportionally. For example, the bit reduction from the FPU32 to the FPU 16 reduces the memory size by 50% (1,880/3,760).

Table 5-5 Memory size

	*Weights Memory (bits)	KBbits	% vs. FPU32
FPU32	3,849,600	3,760	
FPU24	2,887,200	2,820	75
FPU20	2,406,000	2,350	63
FPU16	1,924,800	1,880	50
FPU12	1,443,600	1,410	38

*: Weights memory bits = # of bits $\times$[(400 $\times$300)+300]

**: example of FPU 32 = 32$\times$[(400 $\times$300)+300] = 3,849,600 bits.

It is possible to use the XILINX FPGA 3S4000 which provides the size of 2160Kbits memory (Block RAM: 1728Kbits, distributed memory: 432Kbits) when the FPU16 is necessary.

5-1-3 Power

Power consumption is a very important factor for an embedded system when it is used in a stand alone system.

When considering possible devices for embedded systems and their power comparison, in general, DSP and FPGA use more power than Microprocessors and ASIC (DSP > FPGA > MICOM > ASIC).

XILINX and ALTERA FPGA companies provide the built-in power simulation tool and the FPGA tool. Reports are available after synthesis of the circuits occurs. In our experiment, we used the XILINX Web Power simulation model which is provided on the XILINX website. The parameters which affect power, like the number of nodes and IOs, can be changed on the web site [40].

Bit reduction from the FPU32 to the FPU16 reduces the total power by 14.7% (FPU32: 305mW, FPU16: 261mW) through RAM, Multiplier, and I/O as shown in Table 5-6.

The change of the logic cell does not affect the power at all (refer to CLB of Table 5-6). However, the memory and the multiplier reduction save the power by 4.6%, excluding a number of I/O in an FPGA. In other words, the reduction of the power rate is 4.6% (FPU32 memory and multiplier - FPU16 memory and multiplier) / total power = (26-12)/306 = 4.6%.

In fact, the number of I/Os can be excluded even though it affects power more than those factors, memory, and a hardwired IP (multiplier) because there is no real physical connection pin with the other chips outside.

Table 5-6 Power consumption for different FPUs [unit: mW]

	*CLB	RAM (Width)	Multiplier (Block)	I/O	Sub sum (**Total power)	Decreasing rate (of total power)
FPU32	2	17 (36)	9 (5)	67	95 (306)	
FPU24	2	17 (36)	7 (4)	49	75 (286)	6.5
FPU20	2	17 (36)	4 (2)	45	68 (279)	8.8
FPU16	2	8 (18)	4 (2)	36	50 (261)	14.7 ((306-261)/306)
FPU12	1	8 (18)	4 (2)	29	42 (253)	17.3

*: Configurable logic block (CLB)

**: Total Power = Sub sum+211mW (basic power consumption of the FPGA)

Figure 5-2 shows the power of multiplier and memory in FPGA. Graph is drastically altered at FPU16 and FPU20 because the “block unit” component in the memory and

multiplier IP are used for FPGA. For example, FPU20 and FPU32 use 2~5 block (4~9 mW) multiplier and 36bit width (17 mW) memory caused almost the same power consumption (21~26 mW).

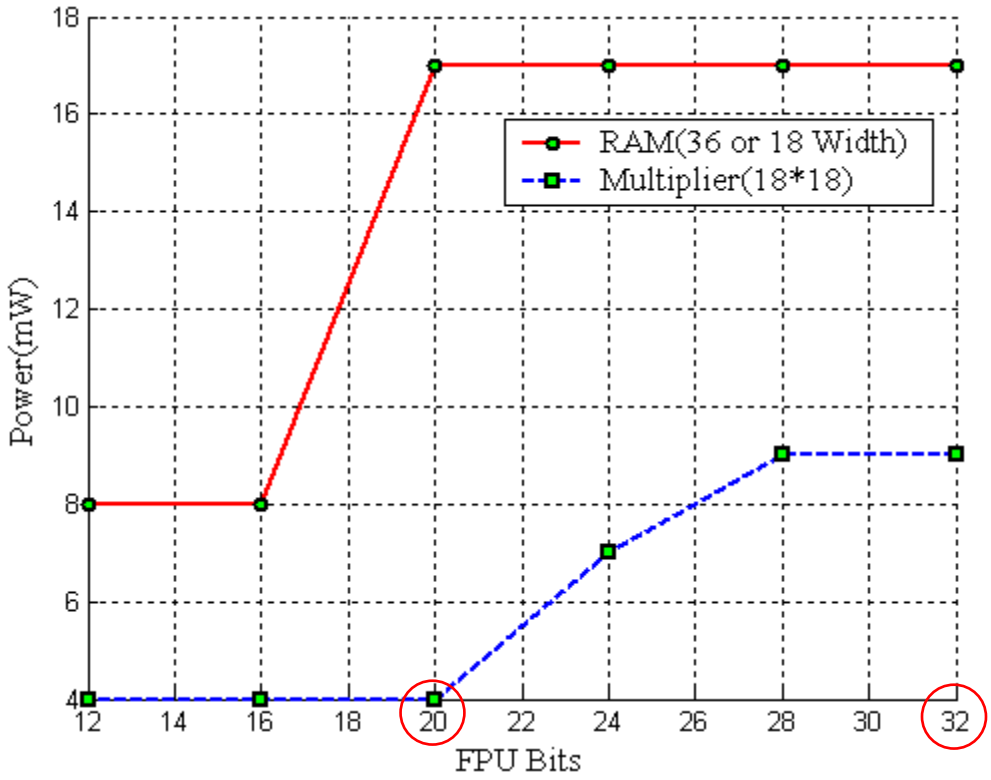
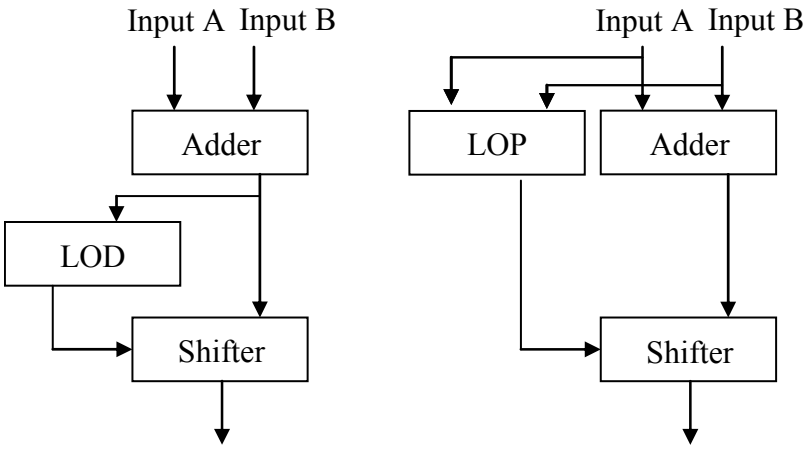


Figure 5-2 Power effect of finite FPU the NN

5-1-4 Architectures of FPU Adder

An NN system and the FPU hardware performance are greatly affected by the FPU addition. The bit-width reduced FPU addition circuit is modified from the commercial IP, LEON. LEON is used as standard architecture [38]. The system performance and clock speeds can be improved by using the leading one prediction (LOP) and close-and-far algorithm [31, 41, 42]. The leading one detection (LOD) for normalization, and rounds

work together at the same time because there is no relationship between two or more processes, as shown in Figure 5-3.



(a) LOD (b) LOP

Figure 5-3 LOP in FPU

The close-and-far algorithm can save the processing time by splitting the circuit into two paths. If a circuit is split by the circuit to two parts, one for a fast calculation and the other one for a slow calculation, the processing time is increased more than twice. According to the studies, 43% of floating-point instructions have an exponent difference of either 0 or 1. A leading-one detector or predictor is needed to count the leading number of zeros only when the effective operation is subtraction and the exponent difference is 0 or 1. For all the other cases, no leading zero count is needed. Following these concepts, the circuits can be divided to two paths. Table 5-7 shows the steps for two paths in addition.

Table 5-7 The steps in the close-and-far cases [31]

	Close	Far
1	Predict exponent	Subtract exponents
2	Subtract significands Predict number of leading zeroes	Align significands
3	Post-normalization	Subtract significands
4	Rounding	Rounding

In our experiments, the FPU adder based on the LOP technique increases the maximum clock frequency by 143% (102.04/71.5). The FPU adder based on the close-and-far algorithm [41, 42] increases the area by 200%, but improves the maximum clock frequency by 280% (200/71.5) as shown in Table 5-8.

Table 5-8 Comparison of different FPU adder architectures (5 pipeline stages)

	LOP	Far-and-close	LEON
Slice	570	1026	486
FF	294	128	269
LUT	1052	1988	905
Min period (ns)	9.8	5	13.98
Max Freq (MHz)	102.04	200	71.5

(Device: SPARTAN, Condition: ISE 6.3i, default)

5-1-5 Hardware Specification Summary

The specification of the implemented FPGA-based face detector is summarized in Table 5-9.

Table 5-9 Specification of a FPGA-based face detector

	Specification
Bit Width	32~16
Frequency	48/80MHz (FPU32 NN/FPU16 NN)
Gates	160,000/120,000 (FPGA SPARTAN) (FPU32/FPU16)
Arithmetic Units	IEEE 754 single precision (Multiplication and Addition)
Networks	2 Layer(400/300/1 node)
Input Data Size	20×20 (400 pixel image)
Operating Time	8.7~5.3ms/frame (FPU32~FPU16)
Frame Rate	114~190/seconds (FPU32~FPU16)

5-2 Algorithm Performance: Detection Error Changes Caused by Activation Function Estimation and Bit-Width Reduced FPU

Two factors affect the detection rate error. One factor is the polynomial estimation error which occurred when an activation function is estimated through the polynomial equation. Another possible error is the error caused by reduced precision of the FPU.

5-2-1 Detection Rate Change by Activation Function Estimation

To reduce the error caused by the polynomial estimation, the polynomial equation can be more elaborately modified as shown in (5.1). The problem of (5.2) is that the function is not differentiable at ± 1, and also, the error (equation (3.14)) will be identically 0 ($f'(sum) = (\pm 1)' = 0$) for $|sum| > 1$. This makes error analysis difficult.

$$f(sum) = 0.75 \times sum \tag{5.1}$$

$$f(sum) = 0.75 \times sum, \ \ |sum| < 1,$$

$$= 1, \ \ sum \geq 1,$$

$$= -1, \ \ sum \leq -1. \tag{5.2}$$

Therefore, the simplified polynomial equation (5.1) was used in this thesis because we are interested in the error caused by the reduced precision FPU. It is confirmed through MATLAB simulation that this polynomial approximation resulted in an average 4.9 % error in the detection rate, as compared to the results of the equation (4.1) in our experiment, as shown in Table 5-10.

Table 5-10 Detection rate of polynomial estimation (MATLAB)

threshold	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	10
tansig	34.09	34.55	37.27	45.91	53.636	61.36	73.09	77.73	75	72.73
Poly ($0.75 \times sum$)	35	39.09	45.91	53.64	62.727	70	72.27	77.27	78.18	77.27
abs diff	0.91	4.54	8.64 (max)	7.73	9.091	8.64	0.82	0.46 (min)	3.18	4.54
Average error	4.9									

5-2-2 Detection Rate Change by Reduced Precision FPU

Table 5-11 shows the detection rate error caused by reduced precision FPUs.

Detection rate includes face and non-face detection rate and average detection rate error is obtained by

|detection rate of FPU 64(PC) – detection rate of reduced precision FPUs|.

The detection rate is changed from FPU32 to FPU16 by 5.91% (72.27-78.18) where the threshold is 0.7. The detection rate is also changed by 5.91% on average.

Therefore, if the FPU16 is used instead of the FPU32, the system will work with 5.91% error on an average in detection rate, and it will be better to use the over 20 bits FPU to reduce the error.

Table 5-11 Detection rate of reduced precision (VHDL)

Threshold	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	Average Detection Rate Error
FPU64 (PC)	35	39.09	45.91	53.64	62.73	70	72.27	77.27	78.18	77.27	
FPU32 NN	35	39.09	45.91	53.64	62.73	70	72.27	76.82	78.18	77.27	0
FPU24 NN	35	39.09	45.91	53.64	62.73	70	72.27	76.82	78.18	77.27	0
FPU20 NN	35	39.09	46.36	53.64	63.18	70	73.64	76.82	77.73	76.82	0.36
FPU18 NN	35	41.36	47.73	56.82	65.46	69.55	74.55	77.73	77.27	74.09	1.73
FPU16 NN	35.91	44.55	53.18	66.36	70.46	76.36	78.18	74.55	72.73	72.73	5.91
|FPU64-FPU16|	0.91	5.45	7.27	12.73	7.73	6.36	5.91	2.73	5.46	4.55	5.91

Figure 5-4 shows the effect of error caused by finite precision arithmetic, the bit-reduce FPU (from left: 18, 16, 14, 12 FPU bits). This result was obtained from MATALB analysis of MODELSIM result file for the FPGA based NN. As FPU bits are reduced to 12bits, the NN result data are collapsed. From this visible result, FPU 18 or more bits FPU is valuable to use.

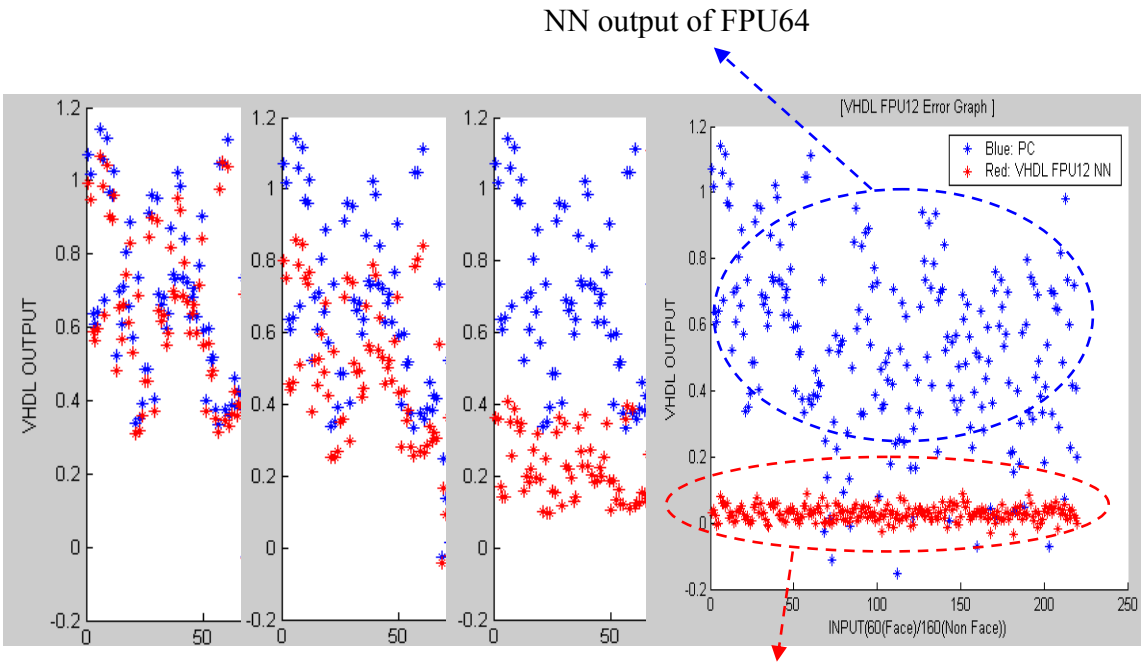


Figure 5-4 NN Output Changes Caused by Bit-Width Reduced FPU

Section 5-2-3 explains how the error of the NN is estimated by reduced precision and detection rate error before implementing the NN in the FPGA.

5-2-3 FPU Bit Decision Graph

Table 5-12 and Figure 5-5 show the output error (|the NN output on PC- the output of VHDL|). Figure 5-6 is the Log graph (base is 2) of those results. Readability is a strong point of a log graph. The readability is better for analysis, and it helps us to decide how many bits are required.

Analytical results are found to be in agreement with simulation results as shown in Figure 5-6. The analytical MRRE results and the maximum experimental results show conformity in shape. The analytical ARRE results and the experimental results (Mean) show conformity in shape. As the n bits in the FPU are reduced within the range from 32 bits to 14 bits, the output error is incremented by 2^n times. For example, 2-bit reduction

from the FPU16 to the FPU14 makes 4 times (2^2) the error. Due to the small number of fraction bits (for example, 5 bits in the FPU12), there was no meaningful result under 14 bits. Therefore, at least 14 bits should be employed to achieve an acceptable detection rate. Refer to Figures 5-5 and 5-6.

Table 5-12 Results of output error on the face detector

	MATLAB		VHDL
	Calculation (MRRE)	Calculation (ARRE)	Experiment (max)
FPU32	4E-05	2.89E-05	1.93E-05
FPU24	0.0026	0.0018	0.0012
FPU20	0.0410	0.0296	0.0192
FPU18	0.1641	0.1184	0.0766
FPU16	0.6560	0.4733	0.2816
FPU14	2.62	1.891	0.9872
FPU12	10.4	7.5256	1.0741

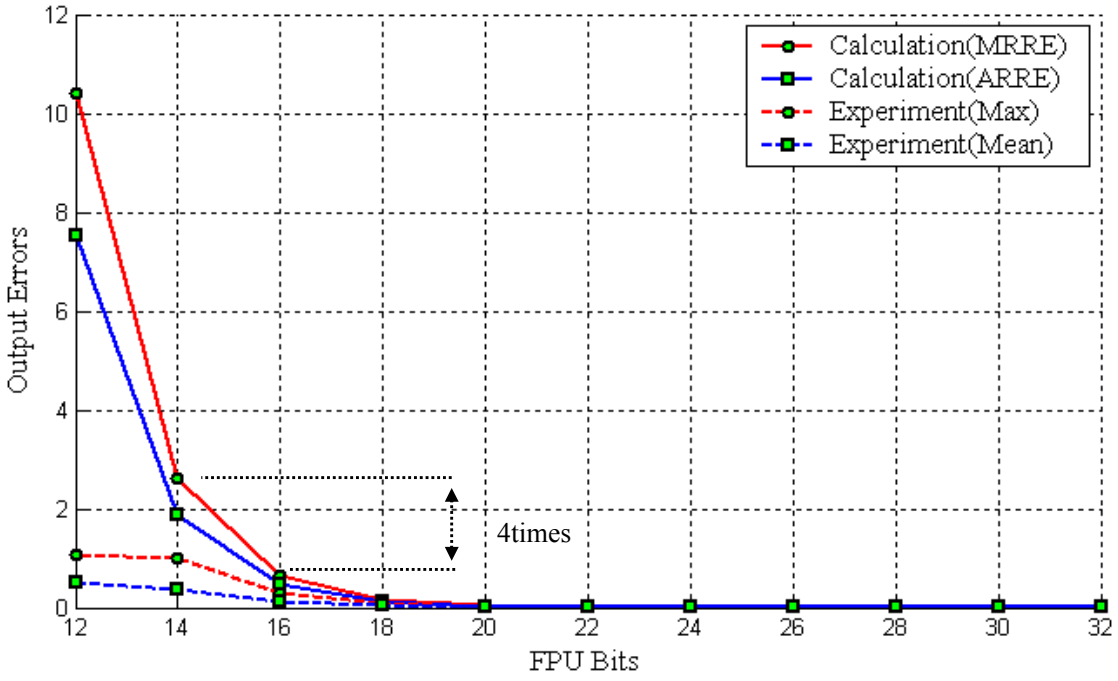


Figure 5-5 Comparison between analytical output errors and experimental output errors

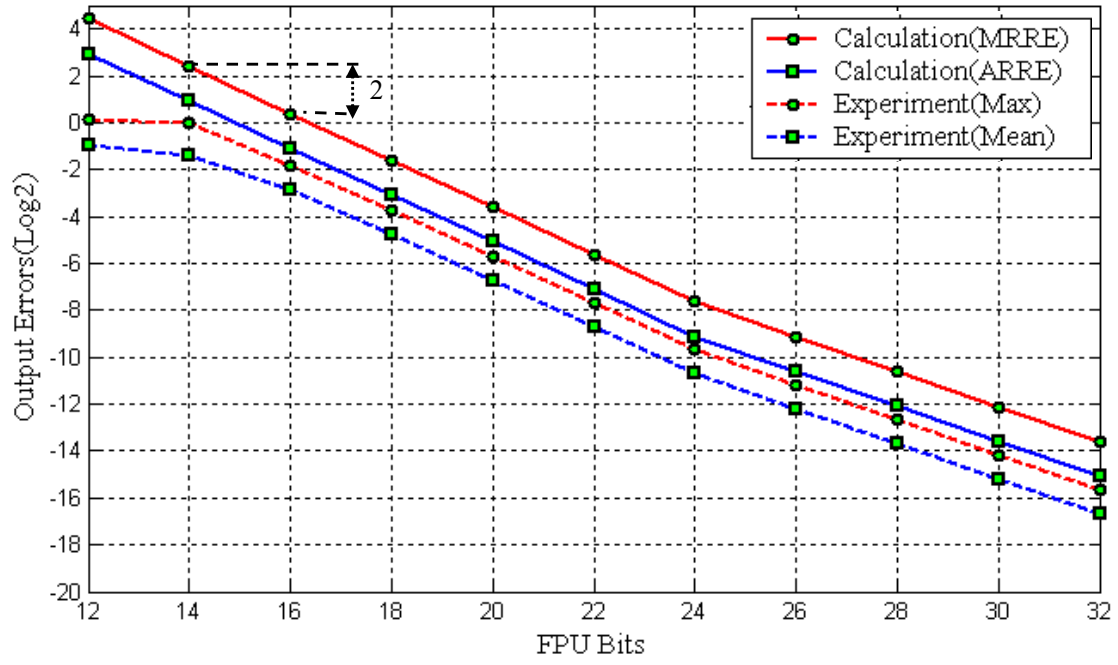


Figure 5-6 Comparison between analytical output errors and experimental output errors (Log_2)

A Signal-to-Noise Ratio (SNR) can be defined to quantify noise by (5.3),

$$SNR = 10\ Log_{10}(P_S/P_N) \tag{5.3}$$

For example, 1 bit reduction creates -3dB error, also called noise,

$$10\ Log_{10}(1/2^{-(-1)}) = -3dB.$$

In another example, a 16 bit reduction arising from conversion from the FPU32 to the FPU16 creates 42dB noise,

$$10\ Log_{10}(1/2^{-(-16)}) = -48.1648dB.$$

5-2-4 Comparison Between Fixed-Point and Floating-Point

In the case of the hardware performance, the FXU is faster than the FPU in clock speed. However, the FPU MAC calculation is known to be more efficient than the FXU in terms of area [43]. Therefore, the FXU is not always better than the FPU in terms of hardware performance. Table 5-12 show the formats and examples of the FPU and the FXU.

The FPU has a smaller size precision and wide range, but the FXU is better than the FPU in terms of accuracy if the FXU precision is smaller than the required precision. Note that the range of the FXU is very small, so that the FXU has many more error possibilities by overflow and underflow than the FPU. Therefore, if the information about range or data bounds are not given (especially, over a large range like in the learning process), the FPU is much more preferable than the FXU in terms of error. It is difficult to estimate the range for the NN because over the course of learning the model, weights and sum data vary over a large range. Therefore, the bit-width reduced FPU is the better arithmetic model for an NN with limited hardware resources.

Table 5-13 Example format of Bit-width reduced FPU and FXU

		Format (S/E/F)	Precision		Range
			Highest	Lowest	
FPU	32	1/8/23	$2^{-23-126}$	$2^{-23+127}$	$\pm[\ 2^{-126} \sim (2-2^{-23})2^{127}\ ]$ $\pm(1.1755e\text{-}38 \sim 3.4028e+38)$
	24	1/6/17	2^{-17-30}	2^{-17+31}	$\pm[\ 2^{-30} \sim (2-2^{-17})2^{31}\ ]$ $\pm(9.3132e\text{-}10 \sim 4.2950e+9)$
	20	1/6/13	2^{-13-30}	2^{-13+31}	$\pm[\ 2^{-30} \sim (2-2^{-13})2^{31}\ ]$ $\pm(9.3132e\text{-}10 \sim 4.2947e+9)$
	16	1/6/9	2^{-9-30}	2^{-9+31}	$\pm[\ 2^{-30} \sim (2-2^{-9})2^{31}\ ]$ $\pm(9.3132e\text{-}10 \sim 4.2908e+9\ )$
	12	1/6/5	2^{-5-30}	2^{-5+31}	$\pm[\ 2^{-30} \sim (2-2^{-5})2^{31}\ ]$ $\pm(9.3132e\text{-}10 \sim 4.2279e+9\ )$
Expression example (FPU12)		2^{-30}			0 000001 00000
		$(2-2^{-5})2^{31}$			0 111110 11111
		±0			0 or 1 000000 00000
		NAN			d 111111 ddddd
		±Infinity			0 or 1 111111 11111

- Bias =31 (1~63) for FPU (exponent=6), bias =127 (1~255) for FPU (exponent=8)
- 2 numbers (all 0 and all 1) is used to express “zero (0)”, “Not a number (NAN)” and “Infinity”.
- The de-normalized numbers using the only fraction bit was not considered.
- d: don’t care, 0 or 1.

		Format *(S/I/F)	Precision (Error)	Range
FXU (least bits constitution)	27	1/3/23	2^{-23}	$\pm[\ 2^{-23} \sim (6+1-2^{-23})\ ]$ $\pm(1.1921e\text{-}7 \sim 7)$
	21	1/3/17	2^{-17}	$\pm[\ 2^{-17} \sim (6+1-2^{-17})\ ]$ $\pm(7.6294e\text{-}6 \sim 7)$
	17	1/3/13	2^{-13}	$\pm[\ 2^{-13} \sim (6+1-2^{-13})\ ]$ $\pm(1.2207e\text{-}4 \sim 6.9999)$
	13	1/3/9	2^{-9}	$\pm[\ 2^{-9} \sim (6+1-2^{-9})\ ]$ $\pm(0.002 \sim 6.9980)$
	9	1/3/5	2^{-5}	$\pm[\ 2^{-5} \sim (6+1-2^{-5})\ ]$ $\pm(0.0313 \sim 6.9688)$
Expression example (FXU9)		2^{-5}		0 000 00001
		$(6+1-2^{-5})$		0 110 11111
		0		0 000 00000
		±Infinity		0 or 1 111 11111

- *: Sign/Integer/Fraction

CHAPTER 6 CONCLUSION

The FPGA face detector was implemented using an NN. To verify the circuit, first of all, the NN was designed and verified on the PC using MATLAB. Next, the weights and test bench data were saved to verify the VHDL code. After the simulation of the NN, area and operating speed were obtained by synthesis using ISE: XILINX FPGA tool. The FPU uses the same calculation method as the PC so that it is easy to verify and easy to change the NN's structure.

It is useful to understand the loss in accuracy and the reduction in costs as the number of bits in an implementation of floating-point representation is reduced. Incremented reductions in the number of bits used can produce useful cost reductions while still meeting any given accuracy requirement.

In this thesis, the analytical error model was developed using the maximum relative representation error (MRRE) and average relative representation error (ARRE) to obtain the maximum and average output errors for the bit-width reduced FPUs. After the development of the analytical error model, the bit-width reduced FPUs and the NN was designed using MATLAB and VHDL. Finally, the analytical (MATLAB) result with the experimental (VHDL) result was compared. The analytical results and the experimental results showed conformity of shape. According to both results, as the n bits in FPU are reduced, the output error is incremented by 2^n times. For example, from FPU14 to FPU16 makes 4 times the output error, $2^{n=16-14=2}=4$. A significantly improved performance was achieved from an FPGA-based face detector implementation using a reduced precision

FPU. For example, it took 5.3 ms in the FPU 16 to process one frame which was 9 times faster than 50 ms (40 ms for loading time + 10 ms for calculation time) of the PC (Pentium 4, 1.4 GHz). Therefore, the number of detectable frames in 1 second increases substantially so that more frames create more opportunities to improve performance. This happens through image enhancement processing with larger images, and to take better pictures, like a trembling compensation effect. For a face recognition system, fast detection performance allows the system to take better pictures and to reduce the burden on the main PC. This high speed performance is also useful to detect car license plates and could be used in military applications that require high speed detections.

It was found that bit reduction from the FPU32 (bits) to the FPU16 (bits) reduced the size of memory and arithmetic units by 50% and the total power consumption by 14%, while still maintaining 94.1% detection accuracy.

Both Bit-width reduced FPU and FPGA are suitable for fast detection systems due to their high speed. We intend to expand the results of this thesis to applications that require high speed detection such as missile and car surveillance systems.

REFERENCES

[1] Face Recognition Vendor Test (FRVT), Available: http://www.frvt.org/FRVT2002

[2] Ming-Hsuan Ming and Yang Hsuan Yang, "Recent Advances in Face Detection," *IEEE International Conference on Pattern Recognition (ICPR) Tutorial*, Cambridge, United Kingdom, Aug 2004.

[3] Ming-Hsuan Yang, David J. Kriegman and Narendra Ahuja, "Detecting Faces in Images: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 34-58, Jan 2002.

[4] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing*, Addison-Wesley, 1992

[5] Ramchan Woo, Sungdae Choi, Ju-Ho Sohn, Seong-Jun Song, Young-Don Bae and Hoi-Jun Yoo, "A Low-Power Graphics LSI integrating 29Mb Embedded DRAM for Mobile Multimedia Applications," *Asia South Pacific design automation Conference (ASP-DAC)*, Yokohama, Japan, pp. 533-534, 2004.

[6] Yongsoon Lee and Seok-Bum Ko, "FPGA Implementation of a Face Detector Using Neural Networks," *IEEE Canadian Conference on Electrical and Computer Engineering*, Ottawa, Canada, pp. 1883-1886, May 2006.

[7] Scott Hauck, "The Future of Reconfigurable Systems," *Keynote Address of 5th Canadian Conference on Field Programmable Devices*, Montreal, Canada, June 1998

[8] H.K. Brown, D.D. Cross and A.G. Whittaker, "Neural Network Number Systems," *International Joint Conference on Neural Networks (IJCNN)*, San Diego, USA, vol. 3, pp. 903-907, June 1990.

[9] Hiroomi Hikawa, "Pulse Mode Multilayer Neural Network based on Floating Point Number Representation," *IEEE International Symposium on Circuits and Systems (ISCAS)*, Geneva Switzerland, vol. 3, pp. 145-148, 2000.

[10] FRIDGE (Fixed-point pRogrammIng DesiGn Environment), Available: http://www.iss.rwth-aachen.de/Projekte/Tools/FRIDGE

[11] MATLAB Fixed-Point Toolbox, Available: http://www.mathworks.com

[12] Hardware Design Studio, Automatic VHDL Generation for DSP Systems - by Steepest Ascent, Reduce Fixed Point Algorithm Implementation Times, Available: http://www.entegra.co.uk/vhdl_generation.htm

[13] XILINX, "Floating-Point Operator v1.0," Available: www.xilinx.com/bvdocs/ipcenter/data_sheet/floating_point.pdf, 2005

[14] ALTERA, "Floating-Point Operator Library," Available: www.altera.com/products/ip/dsp/arithmetic, 2006

[15] Jordan L. Holt and Jenq-Neng Hwang, "Finite Precision Error Analysis of Neural Network Hardware Implementations," *IEEE Transactions on Computer*, vol. 42, no. 3, pp 281-290, 1993.

[16] Selcuk Sen, William Robertson and William J. Phillips, "The Effects of Reduced Precision bit Lengths on Feed forward Neural Networks for Speech Recognition," *IEEE International Conference on Neural Networks (ICNN),* Washington, DC, USA, vol. 4, pp. 1986-1991, 1996.

[17] Claire Fang Fang, Chen Tsuhan and Rob A. Rutenbar, "Floating-point Error Analysis based on Affine Arithmetic," *International Conference on Acoustics, Speech, and Signal Processing* (*ICASSP)*, Hong Kong, vol. 2, pp. 561-564, 2003.

[18] Ho-Man Tung, Michael R. Lyu and Irwin King, "Face Recognition Committee Machine," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP),* Hong Kong, vol. 2, pp. 837-840, 2003.

[19] Biometrics Consortium, "BIOMETRICS", Available: http://www.biometrics.org/

[20] Information Access Division (IAD) of National Institute of Standards and Technology (NIST), Available: http://face.nist.gov/

[21] Scott Sanner, AI and Algorithms Software Archive, Available: http://www.cs.toronto.edu/~ssanner/software.html

[22] Christos Christodoulou and Michael Georgiopoulos, *Applications of Neural Networks in Electromagnetics*, Artech House, 2000.

[23] Abhijit S. Pandya, *Pattern Recognition with Neural Networks in C++*, CRC, 1995.

[24] Youngohc Yoon and Lynn L. Peterson, "Artificial Neural Networks: An Emerging New Technique,*" the Association for Computing Machinery of the Special Interest Group on Business Data Processing conference (ACM SIGBDP),* Orlando, USA, pp. 417-422, 1990.

[25] Henry A. Rowley, Shumeet Baluja and Takeo Kanade, "Neural Network-based Face Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, pp. 23-38, 1998.

[26] The Machine Learning Dictionary, Available: http://www.cse.unsw.edu.au/~billw/mldict.html

[27] "Why use activation functions?," Available: http://www.faqs.org/faqs/ai-faq/neural-nets/part2/section-10.html

[28] Matti Tommiska, "Efficient Digital Implementation of the Sigmoid Function for Reprogrammable Logic," *IEE Proceedings of Computers and Digital Techniques*, vol. 150, pp. 403-411, 2003.

[29] ANSI/IEEE Std 754-1985 "IEEE Standard for Floating-Point Arithmetic," The IEEE, Inc., 1985.

[30] XILNX , SPARTAN-3 FPGA Family Complete Data Sheet.

[31] Israel Koren, *Computer Arithmetic Algorithms,* Natick, MA, pp. 53-92, 2002.

[32] Gokul Govindu, L. Zhuo, S. Choi and V. Prasanna, "Analysis of High-performance Floating-point Arithmetic on FPGAs", *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, Nice, France, pp.149-156, April 2004.

[33] Wikipedia, Specificity, Available: http://en.wikipedia.org/wiki/Specificity

[34] Daniel L. Chester, "Why Two Hidden Layers are Better than One," *International Joint Conference on Neural Networks* (*IJCNN),* Washington, DC, USA, vol. 1, pp. 265-268, 1990.

[35] Marco Krips, Thomas Lammert and Anton Kummert., "FPGA Implementation of a Neural Network for a Real-time Hand Tracking System," *The First IEEE International Workshop on Electronic Design, Test and Applications,* Christchurch, New Zealand, pp. 313-317, 2002.

[36] G.P.K. Economow, E.P. Mariatos, N.M. Economopoulos, D. Lymberopoulos and C.E. Goutis, "FPGA Implementation of Neural Networks: An Application on Medical Expert Systems," *The fourth International Conference on Microelectronics for Neural Networks and FUZZY Systems,* Turin, Italy, pp. 289-293, 1994.

[37] Kai Hwang, *Computer Arithmetic*, John Wiley & Sons Inc, 1979

[38] LEON Processor, Available: http://www.gaisler.com.

[39] Microsoft, Visual C++ Library, Available: msdn2.microsoft.com/en-us/library/ms724408.aspx

[40] XILINX SPARTAN-3 Web Power Tool Version 8.1.01, Available: http://www.XILINX.com /cgi-bin/power_tool/power_SPARTAN3

[41] Ali Malik "Design Tradeoff Analysis of Floating-point Adder in FPGAs," M.S. thesis, Dept. Electrical Eng., University of Saskatchewan, Canada, 2005.

[42] Ali Malik and Seok-Bum Ko, "Effective Implementation of Floating-point Adder Using Pipelined LOP in FPGAs," *Canadian Conference on Electrical and Computer Engineering*, Saskatchewan, Canada, pp. 706–709, 2005.

[43] X. LI, M. Moussa and S. Areibi, "Arithmetic Formats for Implementing Artificial Neural Networks on FPGAs," *Canadian Journal on Electrical and Computer Engineering,* pp. 31-40, 2005.

APPENDIX

A Error Model for an NN using Bit-Width Reduced FPU

A-1 Numerical Error Model for an NN using Taylor Approximation

From equation (3.4) in Section 3-2-1, the error of the 1st layer, ε_j is given by

$$\varepsilon_j = f(\sum_{i=1}^{n} W_{ij} X_i + h) - f(\sum_{i=1}^{n} W_{ij} X_i) + \varepsilon_+ \tag{A.1}$$

(A.2) and (A.3) are obtained by applying Taylor 1st order approximations (A.2) [15, 16].

$$f(x+h) - f(x) = hf'(x). \tag{A.2}$$

$$\varepsilon_j = h \times f'(\sum_{i=1}^{n} W_{ij} X_i) + \varepsilon_+, \tag{A.3}$$

where $h = \sum_{i=1}^{n} (\varepsilon_{Xi} X_i + \varepsilon_{Wij} W_{ij} + \varepsilon_{Wij} \varepsilon_{Xi})$.

The error of the 2nd layer also can be found the same method as shown in equation (A.4).

$$\varepsilon_k = O^f{}_k - O_k = f(\sum_{j=1}^{m} O^f{}_j W^f{}_{jk}) - f(\sum_{j=1}^{m} O_j W_{jk}) + \varepsilon_+. \tag{A.4}$$

From (A.4) replacing the $O^f{}_j$ and the $W^f{}_{jk}$ to $(O_j + \varepsilon_j)$ and $(W_{jk} + \varepsilon_{Wjk})$

$$\varepsilon_k = f\left(\sum_{j=1}^{m} (O_j + \varepsilon_j)(W_{jk} + \varepsilon_{Wjk}) \right) - f(\sum_{j=1}^{m} O_j W_{jk}) + \varepsilon_+. \tag{A.5}$$

Simply,
$$\varepsilon_k = f(\sum_{j=1}^{m}(O_j W_{jk} + h) - f(\sum_{j=1}^{m} O_j W_{jk}) + \varepsilon_+$$

$$\approx h \times f(\sum_{j=1}^{m} O_j W_{jk}) + \varepsilon_+, \tag{A.6}$$

where
$$h = \sum_{j=1}^{m}(\varepsilon_{Wjk} O_j + \varepsilon_j W_{jk} + \varepsilon_{Wjk}\varepsilon_j).$$

$$\approx \left(\sum_{j=1}^{m}(\varepsilon_{Wjk} O_j + \varepsilon_j W_{jk}) \right) f'(\sum_{j=1}^{m} O_j W_{jk}) + \varepsilon_+ . \tag{A.7}$$

The error equation (A.8) can be generalized for n-th layer, l, in a similar way.

$$\varepsilon_l \approx \left(\sum_{k=1}^{o}(\varepsilon_{Wkl} O_k + \varepsilon_k W_{kl}) \right) f'(\sum_{k=1}^{o} O_k W_{kl}) + \varepsilon_+ . \tag{A.8}$$

A-2 MRRE Expression of the Bit-Width Reduced FPU Error Caused by Truncation Rounding

The number of floating point format can be expressed as (A.9).

$$F = (-1)^s \times (1.f) \times 2^{E-bias} \tag{A.9}$$

Where S, E and f are sign, exponent and fraction in floating point format.

Sign (S)	Exponent (E)	Fraction (f)

Also simply,

$$F = C \times (1.f),\ C = (-1)^s \times 2^{E-bias}. \tag{A.10}$$

Ft is the bit-width reduced fraction caused by the truncation rounding and bounded as in (A.11).

$$F_t(\max) < C \times (1.f - 2^{-ulp}) = C \times (1.f) - C \times (2^{-ulp}). \tag{A.11}$$

From (A.11), $\varepsilon_t(\max)$ can be extracted and limited like (A.12)

$$\left|\varepsilon_t(\max)\right| < \left|C \times (2^{-ulp})\right|. \tag{A.12}$$

From (A.10), $$C = \frac{F}{(1.f)} \tag{A.13}$$

C in (A.12) is replaced by (A.13).

$$\left|\varepsilon_t(\max)\right| < \left|\frac{F}{(1.f)} \times (2^{-ulp})\right|. \tag{A.14}$$

A range of fraction is $1 \le 1.f < 2$.

Therefore, the error $\left|\varepsilon_t(\max)\right|$ is limited by (A.15) and (A.16).

$$\left|F \times \frac{1}{2} \times (2^{-ulp})\right| < \left|\varepsilon_t(\max)\right| < \left|F \times (2^{-ulp})\right| \tag{A.15}$$

$$\varepsilon_t(\max) < \left|F \times (2^{-ulp})\right| = \left|F \times MRRE\right| \tag{A.16}$$

Finally, (A.17) is the MRRE expression used to estimate the error caused by truncation rounding. The sign indicates the direction. The truncation rounding error creates negative error shown as Figure A-1.

$$\varepsilon_t(\max) = -F \times MRRE \tag{A.17}$$

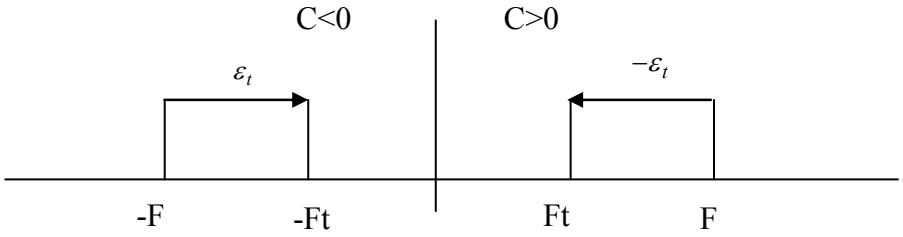


Figure A-1 Error Range of Finite Precision FPU

(A.17) can be used when the ulp is small enough (experimentally, up to FPU14 (ulp = -7)).

The nearest rounding scheme reduces the error by 50% as in (A.18) and creates positive error.

$$\varepsilon_n(\max) = F \times \frac{1}{2} \times MRRE \tag{A.18}$$

A-3 Error Estimation by the MRRE

From (A.3), the 1st error is described by

$$\varepsilon_j \approx \left(\sum_{i=1}^{n} (\varepsilon_W X + \varepsilon_X W) \right) f'(\sum_{i=1}^{n} W_{ij} X_i) + \varepsilon_+ . \tag{A.19}$$

The $\varepsilon_W(\max)$ and $\varepsilon_X(\max)$ terms can be defined by

$\varepsilon_W(\max) = W \times (-MRRE)$ and $\varepsilon_X(\max) = X \times (-MRRE)$.

Thus from (A.19), the error ε_j is bounded so that

$$\varepsilon_j < \left| \left(\sum_{i=1}^{n} [(W_{ij} \times -MRRE) \times X_i + (X_i \times -MRRE) \times W_{ij}] \right) f'(\sum_{i=1}^{n} W_{ij} X_i) + \varepsilon_+ \right|, \tag{A.20}$$

$$\varepsilon_j < \left| -2M \left(\sum_{i=1}^{n} (W_{ij} X_i) \right) f'(\sum_{i=1}^{n} W_{ij} X_i) + \varepsilon_+ \right|, \tag{A.21}$$

Where $\varepsilon_+ \approx \sum_{i=1}^{n} (X_i W_{ij}) \times (-MRRE)$.

If $W, X < 1$ and n is sufficiently large,

$$\varepsilon_j \le \left| 2 \times n \times -M \times f'(\sum_{i=1}^{n} W_{ij} X_i) \right|. \tag{A.22}$$

Finally, the output error of the 2nd layer, ε_k, is also described as shown in (A.13) from (A.6). Where the error of weights can also be written by

$$\varepsilon_{Wjk}(\max) = W_{jk} \times (-MRRE).$$

$$\varepsilon_k < \left| \left(\sum_{j=1}^{m} [(W_{jk} \times -MRRE \times O_j) + (\varepsilon_j \times W_{jk})] \right) \times f'(\sum_{j=1}^{m} O_j W_{jk}) + \varepsilon_+ \right|, \tag{A.23}$$

Where $\varepsilon_+ \approx \sum_{i=1}^{n}(O_j W_{jk}) \times -MRRE$.

If the MRRE ≈ 0, then

$$\varepsilon_k \le \sum_{j=1}^{m}(\varepsilon_j \times W_{jk}) f'(\sum_{j=1}^{m} O_j W_{jk}), \tag{A.24}$$

By using equation (A.21),

$$\varepsilon_k \le \left| \left(2 \times n \times -M \times f'(\sum_{i=1}^{n} W_{ij} X_i) \right) \times \left(\sum_{j=1}^{m}(W_{jk}) \right) \times f'(\sum_{j=1}^{m} O_j W_{jk}) \right|. \tag{A.25}$$

If $W < 1$ and n and m are big enough, then (A.25) can be simplified to (A.26).

$$\varepsilon_k \approx 2 \times n \times m \times -M \times f'(\sum_{i=1}^{n} W_{ij} X_i) \times f'(\sum_{j=1}^{m} O_j W_{jk}). \tag{A.26}$$

B Taylor Series Expansion of the Activation Function

The activation function can be estimated by the Taylor estimation theorem.

Sigmoid function: $\frac{1}{1+e^{-x}}$ (B.27)

By Taylor's theorem, an exponential function can be estimated as in (B.17),

$$e^{x}=\sum_{n=0}^{\infty}\frac{x^{n}}{n!}\approx 1+x+\frac{x^{2}}{2}+\frac{x^{3}}{6}+\frac{x^{4}}{24} \tag{B.28}$$

The 4th order was considered.

Let $\left(1+x+\frac{x^{2}}{2}+\frac{x^{3}}{6}+\frac{x^{4}}{24}\right)$=k, then $e^{-x}=\frac{1}{k}$. (B.29)

$$\frac{1}{1+e^{-x}}\approx\frac{1}{1+\frac{1}{k}}=\frac{24}{48+24x+12x^{2}+4x^{3}+x^{4}} \tag{B.30}$$

Therefore, to design an estimation function, division, multiplication, and addition calculation units are required. This estimation technique is still time consuming; more than over 4 stages of pipelines can be required, moreover, division is a very complicated calculation.

The hyperbolic tangent function can also be estimated in the same way.

Hyperbolic tangent function: $\frac{2}{1+e^{-2x}}-1=\frac{1-e^{-2x}}{1+e^{-2x}}$ (B.31)

In here, $e^{-2x}=\frac{1}{k^{2}}$ using (B.29).

Finally, estimated function is

$$\frac{2}{1+e^{-2x}}-1=\frac{1-\frac{1}{k^2}}{1+\frac{1}{k^2}}=\frac{k^2(k^2-1)}{k^2(k^2+1)}=\frac{k^4-k^2}{k^4+k^2} \qquad \text{(B.32)}$$